BIBLIOTECA BASICA INFORMATICA

BANCOS DE DATOS (I)

21



INGELEK

BIBLIOTECA BASICA INFORMATICA

BANCOS DE DATOS (I) 21

INGELEK

Director editor:
Antonio M. Ferrer Abelló.

Director de producción:
Vicente Robles.

Coordinador y supervisión técnica:
Enrique Monsalve.

Colaboradores:
Angel Segado
Casimiro Zaragoza
Fernando Ruíz
Francisco Ruíz
Jesús Pedraza
Juanjo Alba Ríos
Margarita Caffaratto
María Angeles Gálvez
Marina Caffaratto
Masé González Balandín
Patricia Mordini

Diseño:
Bravo/Lofish.

Dibujos:
José Ochoa.

ISBN del tomo: 84-85831-54-3
ISBN de la obra: 84-85831-31-4
Fotocomposición: Pérez Díaz, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M-4863-1986
Precio en Canarias, Ceuta y Melilla: 380 pts.



INDICE

PROLOGO

El ordenador nos permite básicamente tratar con rapidez un gran volumen de información. El modo más económico de "salvar" los datos es el de escribirlos en disco, tanto si se trata de un disco flexible (disquete o floppy disk) como de un disco rígido (hard disk). Habida cuenta de su estructura, el disco nos proporciona la posibilidad de acceder a los datos de forma directa (o aleatoria) y, por tanto, también secuencial.

El proceso secuencial significa leer o escribir correlativamente, es decir, al comienzo el primer registro, luego el segundo, el tercero, etc., hasta que se acaben los datos. Lamentablemente, en el proceso secuencial si queremos tener acceso a un registro intermedio tenemos que leer todos los registros que le preceden. El proceso directo o aleatorio significa leer o escribir directamente cualquier registro, sin necesidad de procesar los que le preceden. Para que un fichero pueda procesarse de forma aleatoria es necesario, sin embargo, que todos sus registros tengan la misma longitud.

El acceso aleatorio, o directo, sería ciertamente el más versátil si el método de búsqueda fuera el mismo utilizado para cargar los datos. Veremos las técnicas que han ido surgiendo para poder tener un acceso rápido a los datos de formas diferentes a las utilizadas durante su grabación.

El operador dispone de zonas para localizar las informaciones; son los campos ("field"). Cada campo puede contener solamente cifras, letras del alfabeto o bien cifras y letras del alfabeto, signos de puntuación y caracteres especiales. Por supuesto, nor-

malmente no es suficiente un solo campo y por ello utilizaremos varios.

A la unión de todos los campos se le ha dado el nombre de registro. Pero no nos basta un solo registro, sino que utilizaremos muchos. Su conjunto será referido, cuando comencemos a leer o escribir en disco nuestras informaciones, con el extraño nombre de "file" (fichero, o también archivo). Estos ficheros pueden ser tanto de entrada como de salida, es decir, pueden ser leídos o escritos por el ordenador, pero éste puede procesar un único registro a la vez. De ahora en adelante hablaremos de los ficheros de las facturas, de los ficheros de los gastos, de los ficheros de las existencias, etc.

Lo que sigue es precisamente la filosofía de estos campos, registros y ficheros, sobre todo de cómo están organizados, subdivididos (técnicamente se dice "estructurados") y de cómo se pueden manipular sus componentes: eligiéndolos, reuniéndolos, cambiando sus connotaciones, mezclándolos entre sí, etc. Aprenderemos a hacerlo, inicialmente, en los lenguajes BASIC y Pascal. No se trata realmente de un juego: la valiosa información que podemos así manejar, controlada con eficiencia y oportunidad por los ordenadores electrónicos, representa uno de los patrimonios más importantes de los individuos, de las empresas y de la sociedad entera. Es, pues, conveniente que sepamos lo más posible de esta materia.

El término "Banco de datos" está muy generalizado y por eso ha sido empleado como título de este libro; sin embargo, la temática que trataremos sobrepasa lo que, hablando en sentido estricto, es un banco de datos, es decir, la agrupación ordenada de informaciones, para llegar a lo que se puede denominar "base de datos", donde ya se permite controlar y manipular dichas informaciones. La confusión existente en esta terminología (vulgarmente incluso se identifican ambos términos) nos obliga a realizar esta matización antes de comenzar a tratar en profundidad el tema.

CAPITULO I

DENTRO DE LOS FICHEROS

Generalidades

Como acabamos de comentar, cada fichero está constituido por registros (en mayor o menor número, según los casos) compuestos todos ellos por los mismos campos (bloques mínimos de información en una base o banco de datos).

Sin embargo, de un fichero a otro el número, tipo y características de los campos que forman cada uno de los registros puede variar enormemente. Veamos algunos ejemplos.

Imaginemos un fichero que llamaremos FACTURA y que pretende recoger toda la información sobre las facturas emitidas por un pequeño comercio. En él cada registro tendrá como campos de información la fecha de la factura, la mercancía vendida, el comprador, el importe, el IVA, el total y la forma de pago.

Otro fichero, sin embargo, en el que queramos llevar nuestra cartera de proveedores podrá tener como campos los siguientes: empresa, tipo de material que nos suministra, persona de contacto, teléfono y dirección.

Vemos, pues, que, aunque la filosofía sea la misma, la forma de desarrollarla será distinta en cada caso.

Normalmente los registros no son escritos o leídos del disco de forma individual, uno a uno, sino que se reagrupan en bloques, lo cual permite optimizar el proceso de entrada/salida de los datos. El bloque se define así, en general, como el conjunto de caracteres de información que es manejado (leído o escrito) por cualquier dispositivo de entrada/salida.

El bloque puede contener uno o más registros, dependiendo de las dimensiones del registro y de las dimensiones del bloque.

Es de gran utilidad la escritura y la consiguiente lectura en bloques tanto para limitar el número de los accesos necesarios a los dispositivos de entrada y de salida como para optimizar el espacio en disco o en cinta. El control del comienzo y final de bloques se realiza, de forma automática, por el sistema operativo. Por consiguiente, nosotros los usuarios no tenemos que preocuparnos de cómo se efectúa la lectura/escritura y nos bastará saber que:

- durante la grabación, cada vez que ordenamos escribir un registro, es el sistema operativo quien estima el momento oportuno para grabar el bloque de datos correspondiente;
- durante la lectura, cada vez que ordenamos leer un registro, es el sistema operativo quien se encarga de decidir si puede leerlo por completo a partir del bloque que está contenido en memoria o debe leerlo en parte de este bloque y el resto en otro bloque del disco o cinta;
- algunos sistemas operativos, para no desperdiciar espacio, utilizan todos los bytes de cada bloque y, por ello, un registro puede estar "a caballo" entre dos bloques;
- la longitud del bloque varía según el sistema operativo utilizado. Suele ser de 128 bytes o de uno de sus múltiplos (256, 512).

En nuestros programas haremos referencia al fichero por medio de un nombre que se denomina "nombre interno" del fichero. El soporte externo más empleado y cómodo es el disco (bien sea disco flexible, bien disco rígido), por lo que, en lo sucesivo, en lugar de utilizar la expresión soporte externo, emplearemos la expresión "disco". En cada disco se pueden almacenar diferentes ficheros. De aquí la necesidad de distinguir y localizar cada fichero individual.

Cada disco está subdividido en dos partes: una zona está reservada al índice, denominado también DIRECTORIO, y la otra está reservada a los ficheros. El nombre externo del fichero sirve para distinguirlo de los demás ficheros existentes en el disco y es este nombre el que está escrito en el directorio. Los ficheros existentes en disco pueden ser leídos por otros ordenadores, a condición de que utilicen el mismo sistema operativo y el mismo modo de grabación.

Fases de las operaciones con ficheros

Todas las operaciones con los ficheros, sin importar qué sistema operativo se utilice, se desarrollan en tres fases bien distintas: fase de apertura, fase intermedia y fase de cierre.

En la fase de apertura se indica al sistema operativo cuál es el nombre externo del fichero y qué soporte exterior se utiliza; dicho de otro modo, se acopla el nombre interno del fichero con su nombre externo. A partir de este momento el programa, utilizando el nombre interno del fichero, estará en condiciones de hacer comprender al sistema operativo en dónde escribir o leer los datos.

La fase de cierre es muy importante para un fichero en escritura. ¿Recuerda que se tiende a reagrupar varios registros en cada bloque? Según se van juntando y completando bloques se van grabando; al final del programa, es decir, al final de nuestras informaciones, es muy probable que el último bloque no se haya completado, con lo cual es necesario indicar al sistema operativo que puede considerar terminada la operación y que el último bloque debe grabarse tal como está, aunque esté incompleto. Para los ficheros en la fase de lectura no siempre es necesario avisar que se ha acabado de emplear el fichero. No obstante, es buena norma comunicárselo también al sistema operativo; algunos sistemas operativos actualizan el directorio solamente en la fase de cierre.

La fase intermedia es aquella en la que se procede a leer o escribir los registros.

Hemos subdivido este primer volumen dedicado a las bases de datos en dos partes: en la primera parte, por medio de analogías y dibujos, explicaremos algunos tipos de acceso a los ficheros. En la segunda presentaremos ejemplos completos de programas, tanto en BASIC MS DOS como en PASCAL UCSD.

En nuestra exposición trataremos los tipos de acceso siguientes:

- secuencial,
- con índice,
- tipo "hash",
- aleatorio,
- de árbol binario,
- base de datos

Para no hacer demasiado árida la exposición, no hablaremos de algunas técnicas sofisticadas de programación, habida cuenta de que nos interesa, sobre todo, hacer comprender la filosofía de los diversos tipos de acceso.

CAPITULO II

ALGUNOS METODOS DE ACCESO A LOS DATOS

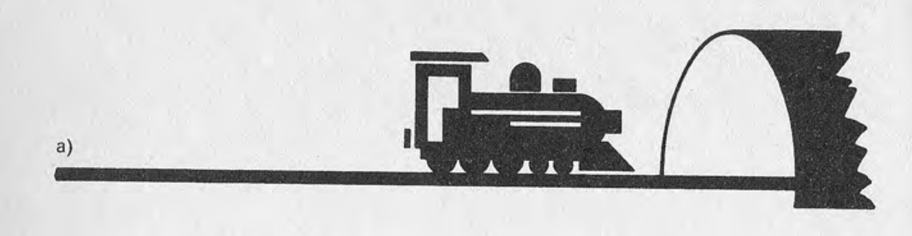
El acceso secuencial

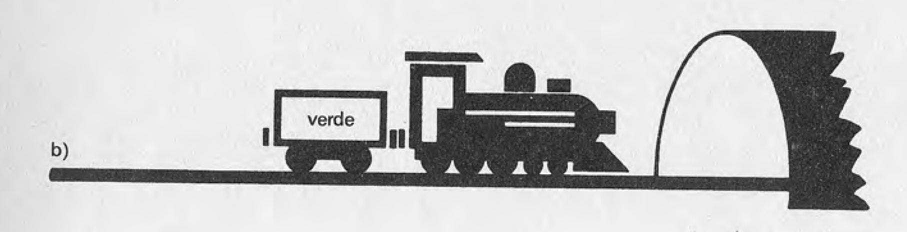
Nuestros ejemplos prevén la utilización de solamente seis registros, cuyos "jefes de familia" respectivos se denominan: rojos, verdes, azules, grises, naranjas y violetas. En nuestras ilustraciones, un tren simulará el fichero y cada vagón corresponderá a un registro. La locomotora es la unidad que se encarga de maniobrar los vagones. Los vagones contienen la información, o sea, los campos. Tiene poca importancia saber cuáles son las informaciones contenidas a excepción de las relativas al primer campo, que es el "padre". En nuestro ejemplo nos interesa que los vagones (los registros) puedan ser tratados en orden alfabético según el padre, aunque las informaciones, como veremos, se nos darán desordenadas.

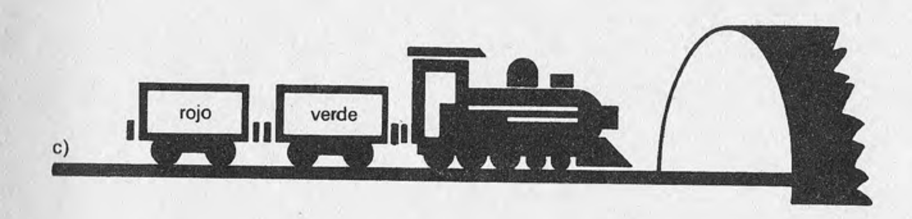
Se habla de acceso secuencial cuando para acceder al contenido de un registro cualquiera (para leerlo o escribirlo) debemos recorrer antes, necesariamente, los que le preceden, cuyo contenido en ese momento no nos importa y al que, además, tampoco tendríamos acceso en ese instante.

En la figura 1 hemos dibujado una locomotora, con la cámara de presión a punto, dispuesta para enganchar vagones y arrastrarlos hacia el túnel que se ve a la derecha. Las vías permiten transportar nuestras informaciones y la locomotora nos permite recoger los vagones y arrastrarlos hacia el túnel. En la figura se muestra con la referencia a) la situación después de la apertura del fichero.

Cuando se quiere utilizar un fichero, la primera operación a realizar es la de abrirlo por primera vez (crearlo); es decir, la que







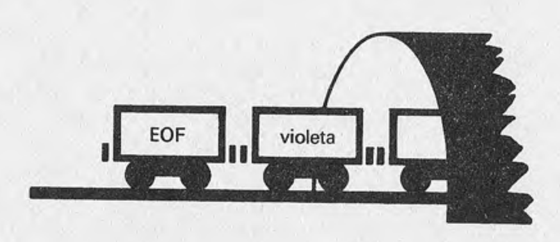


Figura 1.—Creación de un fichero secuencial.

corresponde a solicitar una locomotora que se encargará de trasladar nuestros vagones. Lo que realmente hacemos es indicar al sistema operativo qué unidad ("drive") queremos emplear, qué nombre queremos dar a nuestro fichero y cuál va a ser su estructura. También en los transportes ferroviarios los trenes son referidos mediante una sigla.

Veamos unos ejemplos de apertura de ficheros.

En BASIC MS DOS:

```
500  OPEN 1 AS OUTPUT, "FACTURAS"
510  FIELD 1,32 AS C$10 AS I$20 AS D$6 AS F$
```

En PASCAL UCSD:

```
    VAR TREN1: FICHERO DE REGISTRO
              CLIENTE       : STRING[32] ;
              IMPORTE       : INTEGER[10] ;
              DIRECCION     : STRING[20] ;
              FECHA         : STRING[6] ;
          END;
BEGIN
    REWRITE      (TREN1, '#5:FACTURAS');
```

En ambos casos hemos abierto el fichero FACTURAS especificando el tamaño y nombre de los cuatro campos de cada registro.

En la figura 1 se muestra con la referencia b) lo que sucede con nuestro vagón "verde" después de que lo hayamos cargado con todas las informaciones: simplemente se engancha a la locomotora. Esta última se desplaza hacia la derecha el espacio suficiente para poder enganchar otros vagones.

Veamos ahora cómo podemos escribir en un registro:

En BASIC MS DOS:

```
950  LSET C$ =CL$
960  LSET I$  =MKI$(IMPORTE)
970  LSET D$ =LOCALIDAD$
980  LSET F$ =FECHA$
990  PUT 1
```

En PASCAL UCSD:

```
PUT     (TREN1);
```

Continuaremos así cargando informaciones en los otros vagones y enganchándoles al tren hasta que hayamos completado nuestro fichero. En la figura 1 c) se muestra la situación después de enganchar el vagón "rojo", que es el segundo registro. Hay que destacar el hecho de que cada vagón se ha numerado de forma progresiva a partir del cero. Una vez terminadas las operaciones de escritura (carga) y después de haber enganchado el último vagón, se envía una orden particular con la que se avisa al sistema operativo de que el fichero está completo. El sistema operativo añade entonces al final del fichero una información particular denominada EOF (End Of File - Final de Fichero). En nuestra analogía hemos añadido un pequeño vagón que lleva la indicación de EOF (Fig. 1 d)).

Ejemplo de cierre de ficheros:

En BASIC MS DOS

```
2010 CLOSE 1
```

En PASCAL UCSD:

```
CLOSE    (TREN1,LOCK);
```

En la figura 2 se muestra el posible contenido de un registro. Todos los vagones (registros) deben tener la información dispuesta siempre del mismo modo, que, en nuestro caso, sería: primero, el nombre del cliente, y a continuación, la dirección, la ciudad, el teléfono, el prefijo y el tipo.

Cuando queramos reutilizar nuestro fichero tenemos que abrirlo de un modo muy similar al utilizado para su creación. En nuestra simulación, tenemos que establecer una vía libre en donde se pueda colocar el tren. Como nombre tenemos que darle el mismo utilizado en la creación.

La figura 3 muestra la situación del fichero inmediatamente después de su apertura. Logramos conocer el contenido del primer registro solamente, porque el resto está todavía en las "tinieblas" del túnel. Si queremos tener las informaciones relativas al registro "dorado", por ejemplo, tendremos que hacer avanzar nuestra locomotora hasta que aparezca el vagón correspondiente, pero ya no podremos averiguar de inmediato el contenido de los vagones que pasaron delante.

El acceso secuencial tiene muchas limitaciones. Una de ellas es que para llegar a un determinado registro tenemos que recorrer primero todos los registros anteriores y no nos será posible,



Figura 2.—Posible contenido de un vagón (registro).

además, ver los registros que pasaron con anterioridad; tampoco se podrá añadir ningún registro si no es al final del fichero (antes del EOF); no hace posible la eliminación de un registro. Para subsanar este inconveniente se acostumbra a introducir en cada registro un campo particular que se pone a un valor adecuado si el registro fue objeto de anulación. Como se puede constatar, el control de un fichero secuencial es muy sencillo, pero no precisamente flexible, y ello impone bastantes limitaciones. Es precisamente buscando solventar esta escasa flexibilidad y la lentitud que lleva pareja por lo que se han estudiado otras clases de accesos.

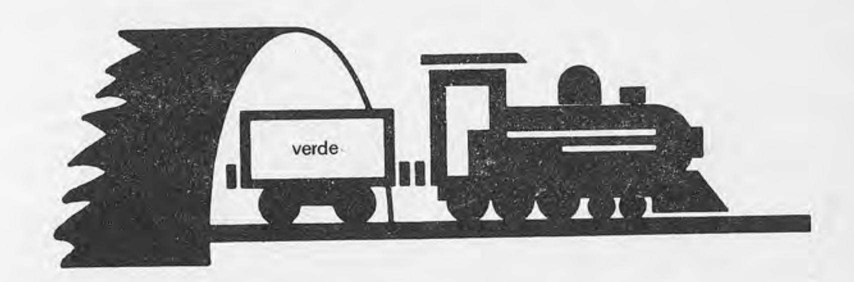


Figura 3.—Lectura de un fichero secuencial: situación después de la apertura.

El acceso directo (o aleatorio)

Se habla de un fichero de acceso directo (o aleatorio) cuando podemos alcanzar el contenido del registro que nos interesa tan sólo indicando su identidad (bien a través de su número o de una clave) y sin tener que recorrer previamente ningún otro.

Para explicar el acceso aleatorio seguiremos utilizando la locomotora y los vagones, pero en lugar de arrancar nuestro tren de la oscuridad del túnel con todos los vagones unidos, colocaremos cada vagón en una vía muerta diferente. Cada vía se numerará, de forma progresiva, desde cero en adelante. Como en el caso del fichero secuencial, se deberá prefijar de antemano una locomotora, es decir, se tendrá que abrir el fichero.

En la figura 4 a) se muestra la situación antes de la colocación del primer vagón (el registro "verde"). La locomotora tiene delante de sí diferentes vías muertas. Su "sentido común" le indica que el primer vagón debe colocarse en la vía 0, el segundo vagón en la vía 1 y así sucesivamente hasta el séptimo vagón, en la vía 6. En la práctica, las vías se ocuparán una tras otra sin dejar ningún espacio vacío.

En la figura 4 b) se ilustra la situación después de escribir el registro "verde" y unos instantes antes de escribir el registro "rojo". El vagón "verde" está ya colocado en la vía 0 y el vagón "rojo", arrastrado por la locomotora, está preparado para llegar a la vía 1.

En la figura 5 se muestra la situación final. La locomotora está inactiva y las vías 0 a la 5 contienen cada una un vagón. La última vía, la que tiene como referencia el número 6, contiene un indicador con el rótulo EOF (End Of File - Final de Fichero) que, en la práctica, señala que no se puede ir más allá. También el cierre del fichero se efectúa de la manera vista para los ficheros secuenciales. Asimismo no es posible eliminar directamente registros; se acostumbra a utilizar la técnica de insertar un campo que contie-

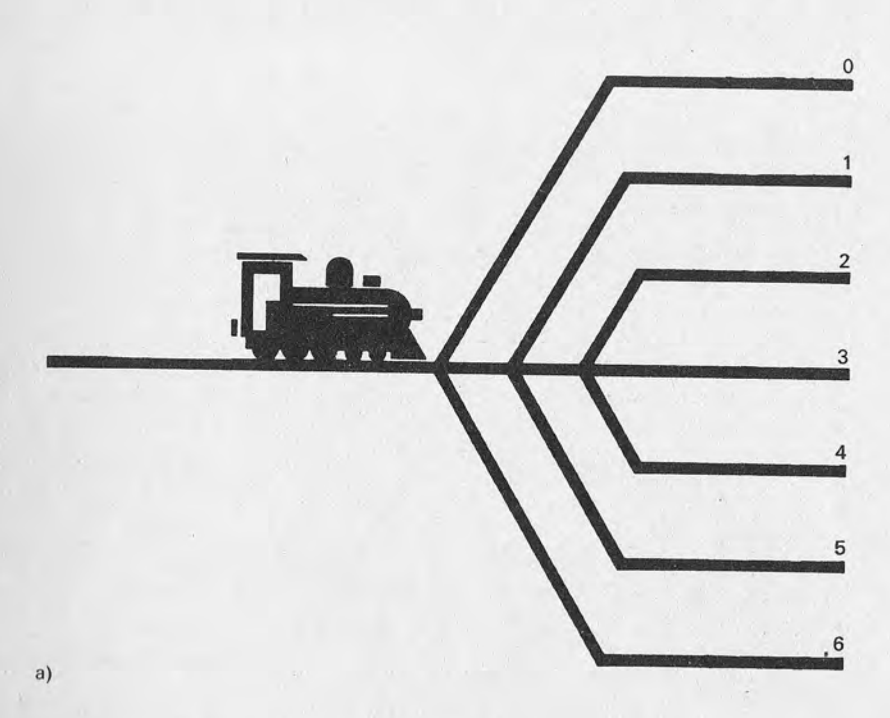


Figura 4.—Creación de un fichero de acceso directo (aleatorio).

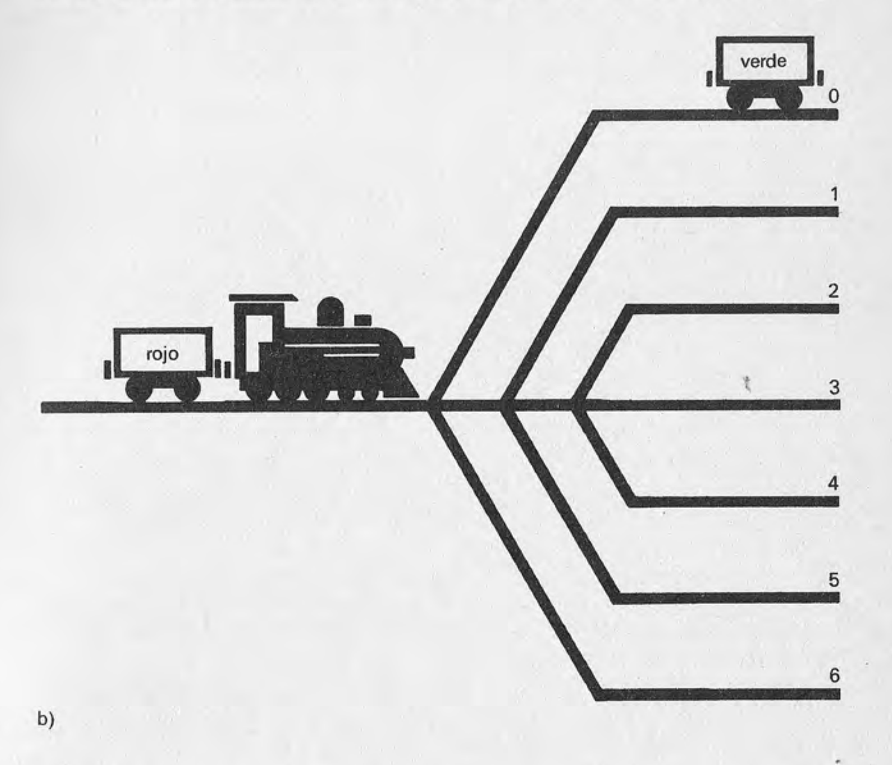


Figura 4.—Creación de un fichero de acceso directo (aleatorio).

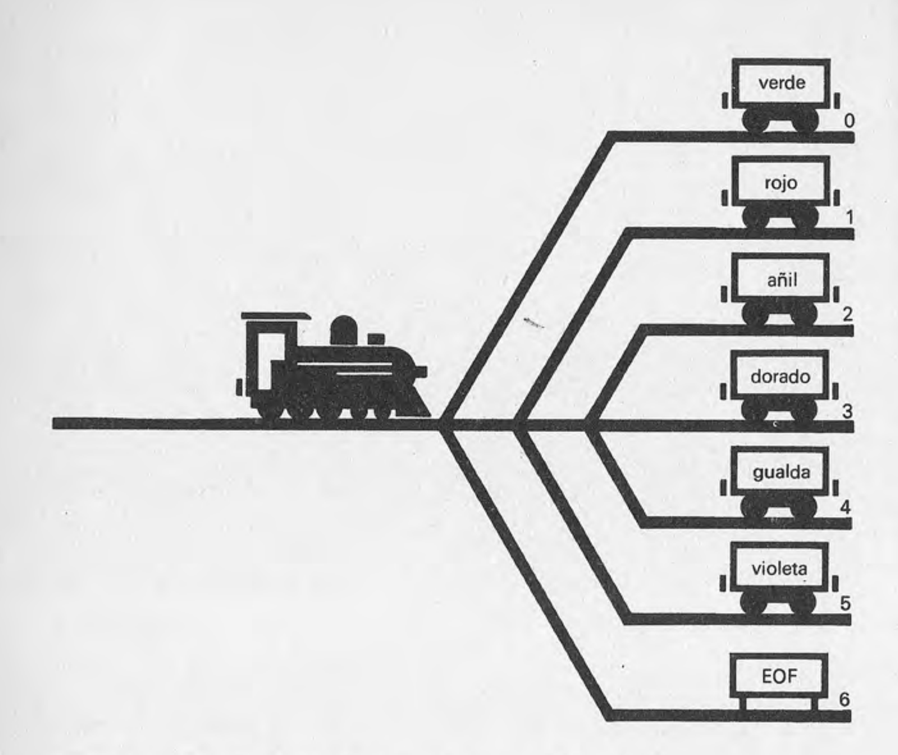


Figura 5.—Situación final, cerrado ya el fichero.

ne un valor particular si el registro se ha anulado. Habrá intuido ya que para tener acceso a un vagón determinado basta saber en qué vía se encuentra, aunque no siempre sea fácil conocer el número de los registros.

En la figura 6 se muestra la locomotora que ha tomado el vagón que se encontraba en la vía 2. No obstante, en principio no disponemos de ningún sistema para saber que en la vía 2 se encuentra el vagón "añil". ¿Qué hace el sistema operativo para determinar la posición de cualquier registro dentro del fichero? ¿Qué hará para determinar en qué bloque está contenido y a partir de qué posición del mismo bloque comienza?

Supongamos que nuestro registro tiene una longitud máxima de 64 caracteres y que los bloques en el disco tienen una longitud de 256 bytes (de 0 a 255). Se tendrá, pues, que:

- el primer registro estará escrito en el primer bloque, a partir de la posición 0;

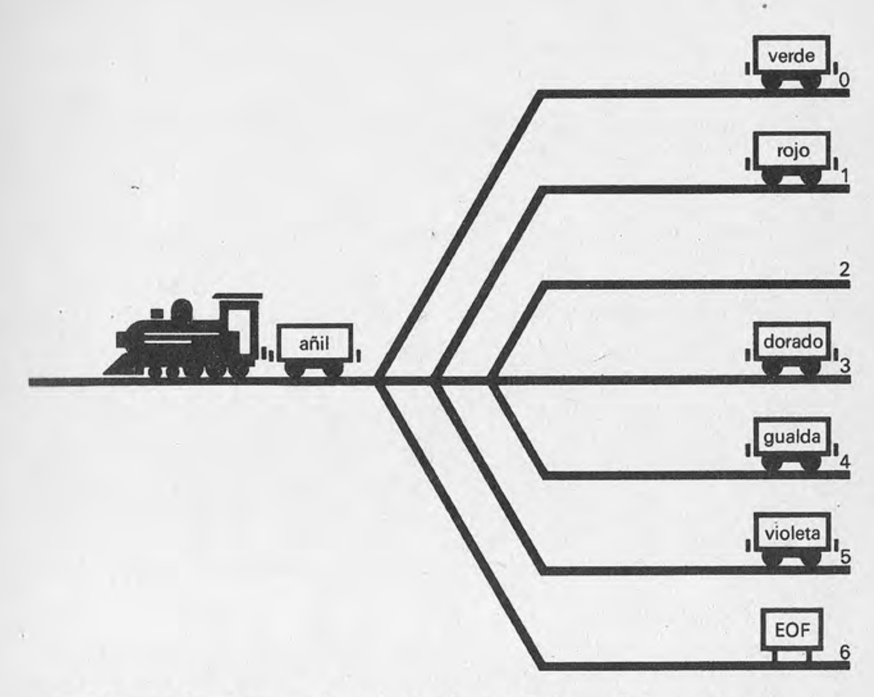


Figura 6.—Acceso al registro número 2 de un fichero directo.

- el segundo registro estará escrito en el primer bloque, a partir de la posición 64;
- el tercer registro estará escrito en el primer bloque, a partir de la posición 128;
- el cuarto registro estará escrito en el primer bloque, a partir de la posición 192;
- el quinto registro estará escrito en el segundo bloque, a partir de la posición 0;
- el sexto registro estará escrito en el segundo bloque, a partir de la posición 64.

Y así sucesivamente.

Si queremos tener acceso a un registro concreto (por ejemplo, el número 100), haremos la siguiente operación:

(nº registro-1)*64

Por ejemplo:

(100-1)*64=6336

Es decir, el registro número 100 (partiendo del 1, no del 0) parte del byte número 6336. Para determinar en qué bloque se encuentra un byte (el 6336) se procederá así:

Parte entera de (nº byte/256) + 1.

INT(6336/256) + 1 = 25

El registro 100 se encuentra en el bloque 25 (partiendo de 1, no de 0), pero, ¿en qué byte? Se tendrá:

Resto de (6336/256) = 192

Por consiguiente, el registro 100 se encuentra a partir del byte 192 del bloque 25.

El acceso directo es muy rápido porque vamos, a ciencia cierta, al registro ("n") buscado. La limitación de este acceso radica en el hecho de que no se sabe qué registro hay en cada lugar (vía). Dicho de otro modo, tenemos que desarrollar otro sistema para saber qué información se encuentra en cada vía.

En este punto, tenemos que introducir un nuevo concepto que es el de campo "clave". Si queremos buscar un registro basándonos en el campo "apellido", se dice que "apellido" es un campo clave; si deseamos buscar un registro sobre la base del campo "mercancía", se dice que "mercancía" es una clave.

Se dice que un campo es un "campo clave" cuando su contenido es usado para encontrar los registros. El campo clave nos permite la búsqueda de un registro mediante su contenido. En nuestros ejemplos la búsqueda se realizaría basándose en el nombre del cliente. Para la búsqueda mediante campos clave veremos dos métodos: el acceso mediante INDICE y el acceso de tipo HASH.

Acceso mediante índice (ficheros directos)

Los diversos registros se almacenan como hemos visto para los ficheros de acceso aleatorio. La locomotora deposita el primer vagón en la vía 0, el segundo vagón en la vía 1 y así sucesivamente, hasta el sexto vagón en la vía 5. Mientras tanto se crea una tabla en donde están almacenadas las claves (fijadas antes) de cada registro individual y en qué posición del fichero están guardadas. Hemos representado esta tabla como una serie de cajas adosadas. Cada caja contiene una clave y la posición relativa del registro, es decir, el número de vía. El número de cada caja, sin

embargo, no es significativo y sirve sólo como referencia en la explicación del proceso que se lleva a cabo.

La figura 7 muestra la situación después de la apertura del fichero. La locomotora está preparada para depositar el vagón de "verde". Las vías y las cajas están vacías.

Nosotros tomaremos como clave de índice el nombre del vagón (verde, rojo...). La figura 8 muestra la situación después de la escritura del registro "verde". El vagón "verde" ocupa la vía 0. La caja 0 contiene esta información: si busca la clave "verde" sabrá que debe retirar el vagón que se encuentra en la vía 0. Los demás vagones se depositarán en la forma habitual, es decir, el segundo en la vía 1, el tercero en la 2, etc. Por el contrario, será diferente el control de las cajas. En ellas, las informaciones deben introducirse en orden alfabético ascendente. Cuando escribamos el registro "rojo" se colocará en la vía 1, pero tenemos que introducir la clave ("llave") de "rojo" en la caja 0, aunque esté ocupada.

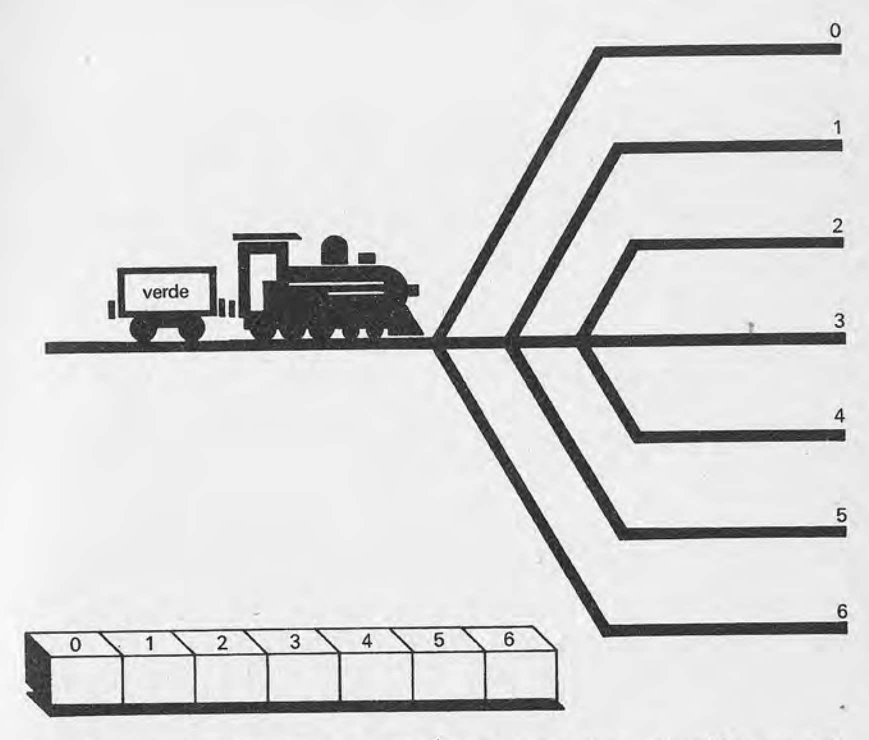


Figura 7.—Fichero con acceso mediante índice: situación antes de la escritura del registro correspondiente a "verde".

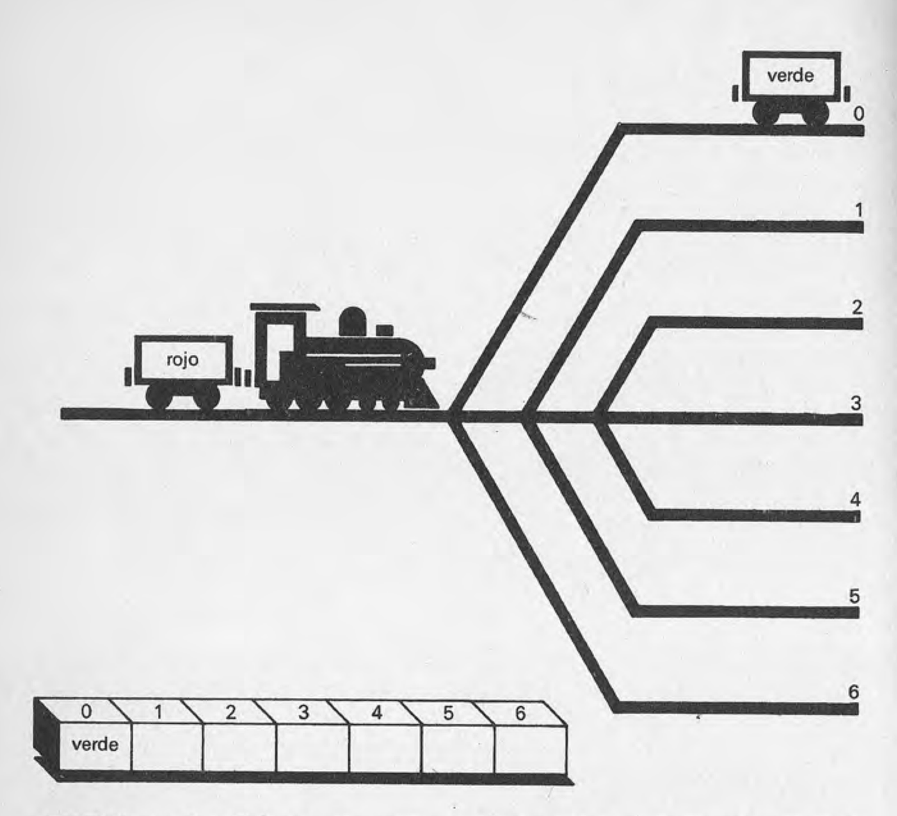


Figura 8.—Fichero con acceso mediante índice: situación después de la actualización de la clave del registro correspondiente a "verde".

Esto es así porque las claves deben estar en orden alfabético y "rojo" debe colocarse antes de "verde". Por consiguiente, tendremos que desplazar el contenido de la caja 0 a la caja 1 antes de introducir en la primera las informaciones de "rojo".

En la figura 9 observamos que en la vía 0 se encuentra el vagón "verde" y en la vía 1 está el vagón "rojo". En las cajas vemos que las claves están en orden alfabético ascendente. Ahora sabemos que el vagón "rojo" se encuentra en la vía 1 mientras que el vargón "verde" está en la vía 0. Debemos convenir en que es mejor desplazar las informaciones en las cajas que no intercambiar entre sí dos vagones. ¡Piense tan sólo en las maniobras que tendría que realizar la locomotora para efectuar el intercambio! La creación prosigue con el depósito de vagones en las siguientes vías libres, mientras que en las cajas se procede a insertar las cla-

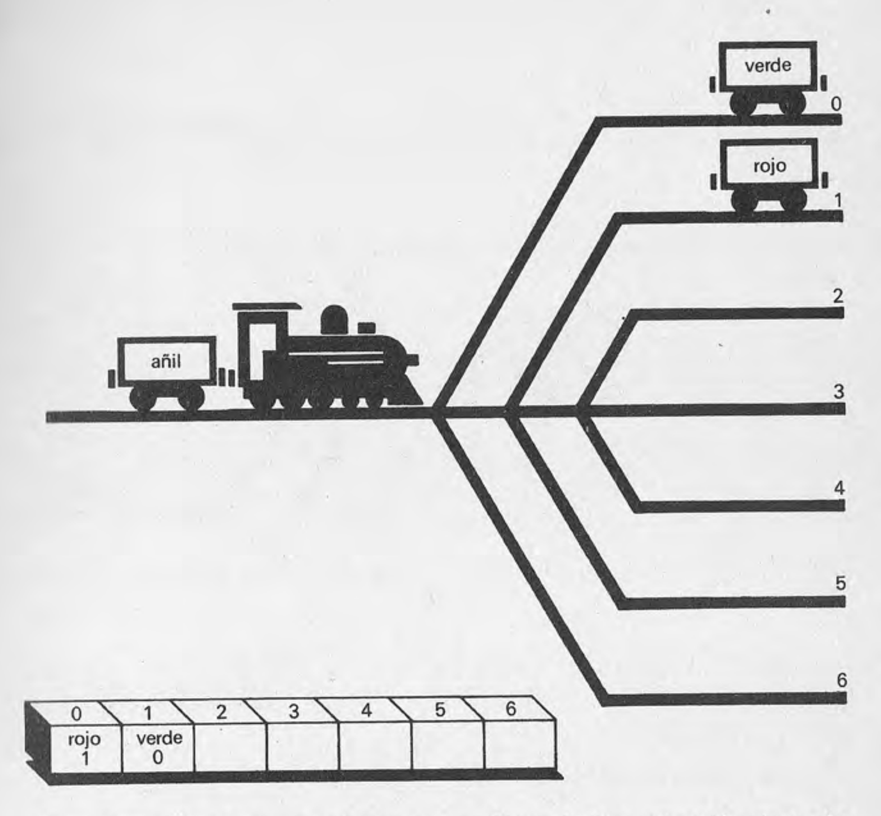


Figura 9.—Fichero con acceso mediante índice: situación después de la actualización de la clave del registro correspondiente a "rojo".

ves (los nombres) y el número de la vía de modo que las claves estén en orden alfabético ascendente.

La figura 10 muestra la situación final y el contenido de las cajas: cada una contiene una clave y el número de la vía correspondiente.

Antes de cerrar definitivamente un fichero con acceso mediante clave o índice, el contenido de las cajas debe guardarse (se deposita de forma ordenada en uno o varios vagones). La figura 11 proporciona una representación esquemática de la composición efectiva final del fichero con acceso mediante índice. En condiciones normales, los vagones de las claves preceden a los demás vagones. En b) está representado el contenido del vagón de las claves.

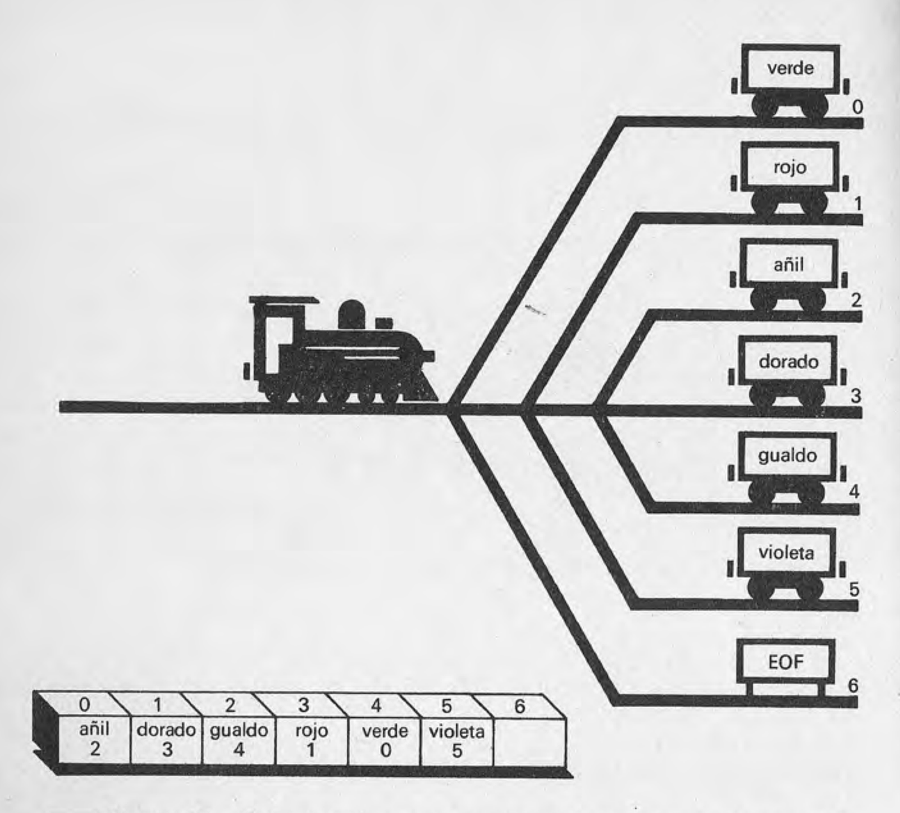


Figura 10.—Fichero con acceso mediante índice: situación después de la grabación de toda la información.

Cómo se accede al fichero

Si queremos tener acceso al registro "rojo" no tendremos más que examinar las cajas de nuestros armarios hasta que encontremos la clave "rojo"; en la misma caja se nos dice que el vagón se encuentra en la vía 1. Para localizar la clave utilizamos una clase de búsqueda que se denomina secuencial: se parte de la primera caja y se ve si contiene la clave buscada; de no ser así, se prosigue la búsqueda en la siguiente. La búsqueda concluye en tres casos: porque hayamos encontrado la clave, porque hayamos consultado todos los compartimentos sin resultado satisfactorio o porque la clave de búsqueda tenga un valor menor que el de las claves siguientes (es como si no existiera ninguna clave).

a)



b)

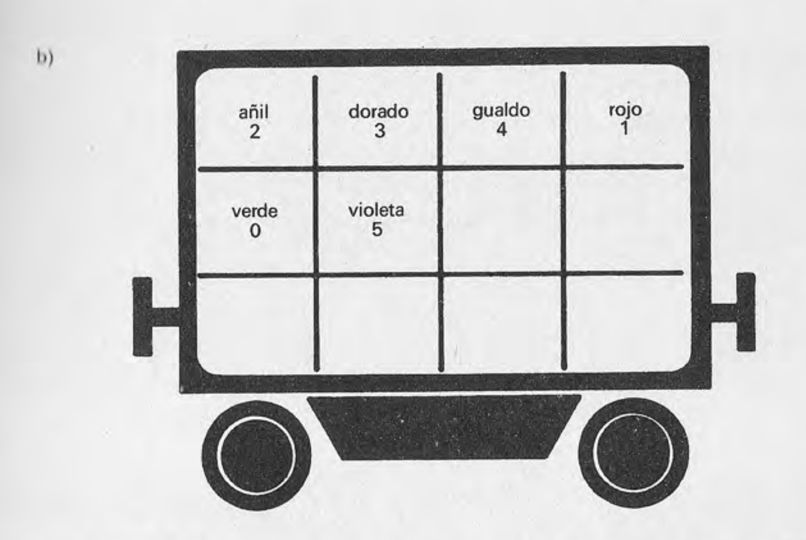


Figura 11.—Fichero con acceso mediante índice: a) El fichero lleva en la cabecera todas las claves. Siguen los registros en el orden de grabación. b) Contenido del pequeño vagón de las claves.

Existen métodos más rápidos para la búsqueda en una tabla ordenada. Uno de los más conocidos es la técnica de la búsqueda dicotómica o binaria, que consiste en: se divide la tabla en dos partes, se compara la clave con el elemento mediano de la tabla. Si se trata del elemento buscado, el proceso está acabado. Si la clave buscada es menor significa que se encuentra en la primera mitad de la tabla y, de no ser así, deberá encontrarse en la segunda. La conclusión es que podemos limitar nuestra búsqueda a solamente una de las mitades de la tabla; por lo tanto, tomamos esta parte y la dividimos también en dos, volviendo a iniciar las comparaciones en la forma anteriormente explicada. Llegamos a un punto en el que se presentan dos casos: encontramos la clave o bien no podemos subdividir más la tabla.

Si se quiere eliminar un registro bastará anular la entrada correspondiente en la tabla de las claves. Con frecuencia se acostumbra a señalar el registro que se ha anulado, con el fin de per-

mitir la reconstrucción del fichero en el caso de su destrucción parcial o para recuperar el espacio ocupado por el registro anulado. La figura 12 muestra la situación después de la anulación del registro "gualdo", pudiéndose observar que en los compartimentos se ha eliminado la clave "gualdo" y las claves se han desplazado un lugar a la izquierda.

Veamos ahora un ejemplo de búsqueda dicotómica en una tabla constituida por varios elementos. Realizaremos búsquedas tanto de elementos que existen como de elementos inexistentes y explicaremos además un interesante recurso que permite seguir paso a paso la lógica del programa y que enseña a encontrar los "gazapos". Presentaremos una tabla de 21 elementos, en donde cada elemento corresponde a una letra del alfabeto, aunque, naturalmente, puede sustituirse por cualquier tipo de cadena, a con-

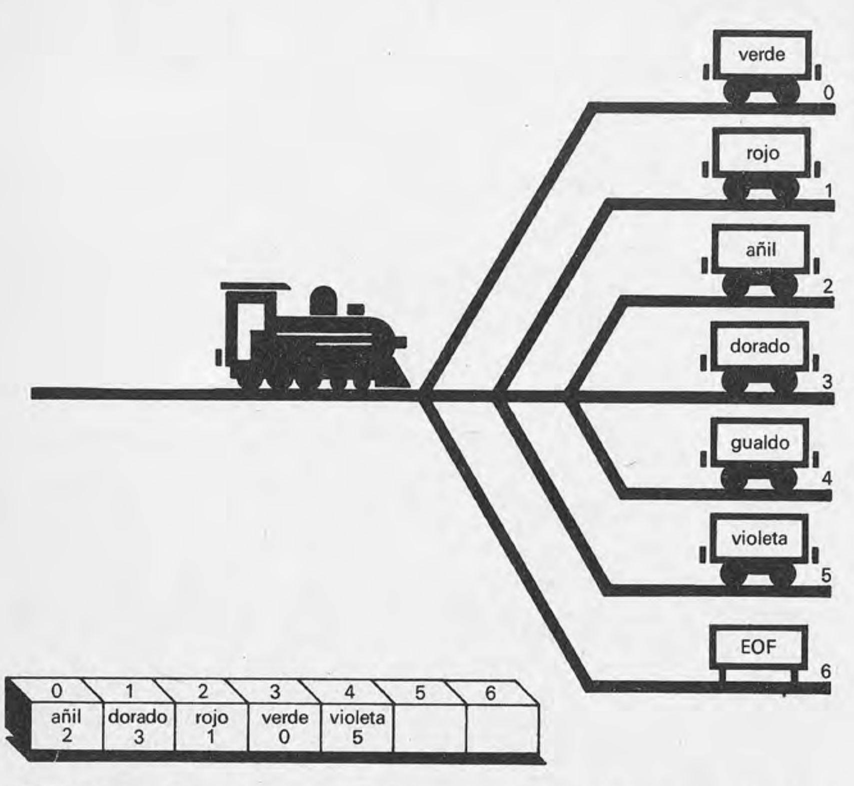


Figura 12.—Acceso a un fichero secuencial de índice: situación después de la supresión del registro "gualdo".

dición de que queden en orden alfabético ascendente. A continuación damos la tabla correspondiente, en donde hemos supuesto también una numeración progresiva que, aunque no estando presente en la tabla, nos hará más cómodas las explicaciones:

1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	L
11	M
12	N
13	O
14	P
15	Q
16	R
17	S
18	T
19	U
20	V
21	Z

Hemos prometido utilizar determinados recursos para explicar mejor todo lo relativo a la búsqueda dicotómica. En cumplimiento de dicha promesa veamos primero la figura 13, que presenta, mediante un diagrama de flujo, la subrutina de búsqueda y, en segundo lugar, veremos la tabla de estado.

La "tabla de estado" es de gran utilidad cuando se tiene que seguir la lógica de una parte de programa observando el contenido de determinadas variables. La tabla, puesto que de ello se trata, está constituida por diversos elementos (columnas). Tendremos tantos elementos como variables. Haremos ver el contenido de los elementos en diferentes momentos y en cada instante se pondrá de manifiesto en una línea. Numeraremos las líneas de forma progresiva. En las primeras ocasiones resultará difícil leer una tabla de estado, pero estamos en condiciones de asegurar que, una vez tomada cierta confianza, será muy útil para las operaciones de depuración.

¿Qué variables destacar? Es evidente que lo primero que queremos ver es el contenido de un elemento de la tabla; en nuestro caso, una letra del alfabeto. Por consiguiente, es conveniente sa-

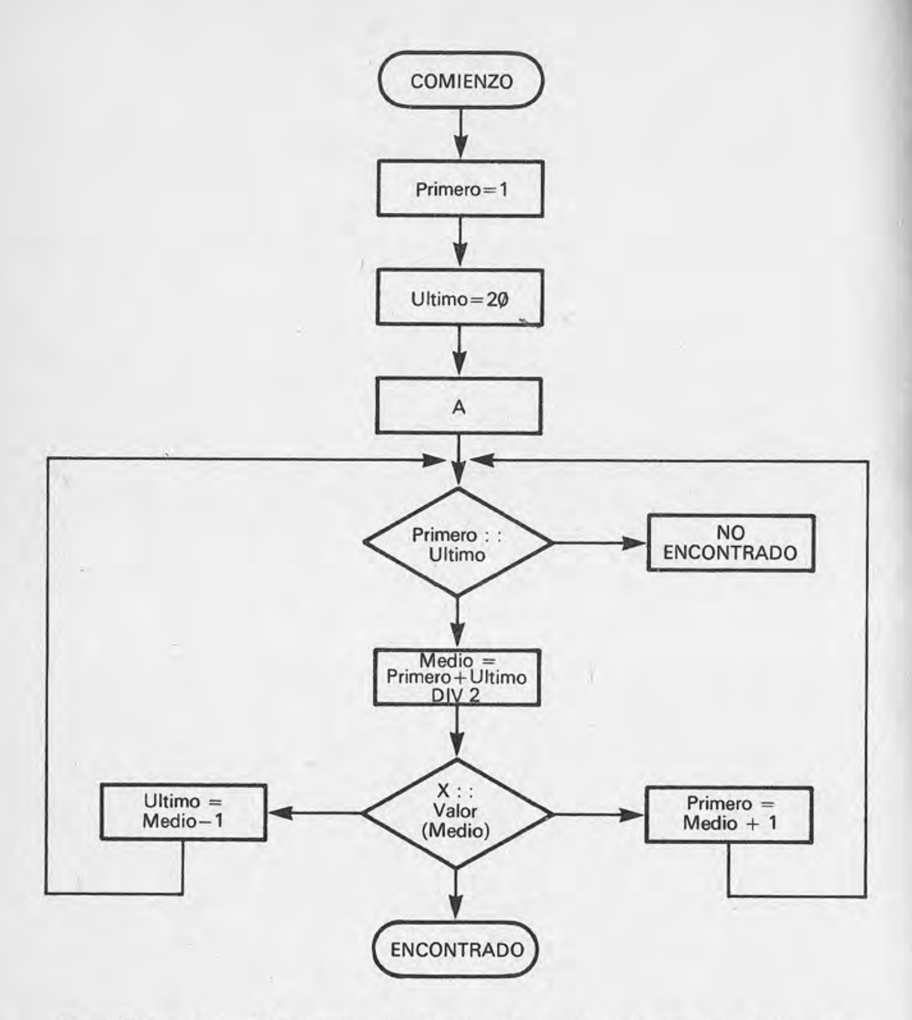


Figura 13.—Búsqueda dicotómica en la gestión de un fichero de índices.

ber la variable que "apunta" al elemento de la tabla; a dicha variable la hemos denominado MEDIO, mientras que llamaremos ULTIMO al último elemento y PRIMERO al primer elemento de la tabla. Las variables PRIMERO y ULTIMO serán modificadas por la rutina de búsqueda (observe la figura 13), al igual que la variable MEDIO. En las tablas siguientes observará que repetiremos las variables PRIMERO y ULTIMO porque tenemos que conocer el valor

que se acaba de introducir en la subrutina y porque algunas instrucciones modificarán su contenido. Las dos columnas repetidas se han encabezado de modo diferente y, desde el punto de vista físico, se han colocado de manera que respeten la situación "inicial" y la situación "final".

Supongamos que se trata de buscar la letra del alfabeto "D". Escribiremos la cabecera de la tabla de estado como sigue:

Paso	Inicial		Medio	Valor	Final	
	Pr.	Ult.		(medio)	Pr.	Ult.

"Inicial Pr." representa el valor inicial de la variable PRIMERO.

"Inicial Ult." significa el valor inicial de la variable ULTIMO.

"Final Pr." contiene el valor final de la variable PRIMERO.

"Final Ult." supone el valor final de la variable ULTIMO.

Bajo el encabezamiento "paso" escribimos 1, en la columna "Primero inicial" escribimos también 1 y bajo "Ultimo inicial" escribimos 20. Estos dos últimos valores los deducimos del flujo del diagrama de la figura 13. El valor medio viene dado por 1+21 y el total dividido por 2. El resultado lo escribimos en la columna "Medio". El elemento 11 de la tabla es M y lo transcribimos a la "Medio". El elemento 11 de la tabla es M y lo transcribimos a la columna "Valor (medio)". Queremos buscar la "D" y esta letra tiene un valor menor que M. Por consiguiente, el valor de ULTIMO deberá modificarse. El nuevo valor de ULTIMO es igual al valor de MEDIO disminuido en 1; es decir, 10. Dicho valor lo escribimos en la columna de "último final". A la columna "primero final" llevamos el contenido de "primero inicial", porque dicha variable no se modificó. La tabla de estado debe presentarse así:

Paso	Inicial		Medio	Valor (medio)	Final	
	Pr.	Ult.			Pr.	Ult.
1	1	21	11	M	1	10

Volvemos al punto "A" y, puesto que el valor de la variable PRIMERO no es mayor que el de la variable ULTIMO, completamos ahora la segunda línea de la tabla de estado. En la columna "paso" escribimos 2 y llevamos a las dos columnas "inicial" los valores de las variables PRIMERO y ULTIMO, obtenidos de las dos últimas columnas de la fila anterior. Por lo tanto, debajo de la variable PRIMERO escribiremos 1 y debajo de la variable ULTIMO escribiremos 10. A continuación calculamos el valor de la variable MEDIO ((1+11)/2) y el valor obtenido, que es 5, se escribe

en la columna de la variable MEDIO (observe que no se tiene en cuenta el resto). Podemos escribir ahora en la columna "Valor (medio)" el valor "E". Puesto que el valor "D" es menor que "E" modificaremos solamente la variable "último" y así, bajo el encabezamiento "último final" escribimos 4 (MEDIO–1) y bajo "primero final" escribimos el valor anterior, es decir, 1.

La tabla de estado debe presentarse como sigue:

Paso	Inicial		Medio	Valor (medio)	Final	
	Pr.	Ult.			Pr.	Ult.
1	1	21	22	M	1	10
2	1	10	5	E	1	4

Volvamos al punto "A". Como el valor de la variable PRIMERO no es mayor que el de la variable ULTIMO, completamos la tercera línea. Bajo el encabezamiento "paso" escribimos 3, y bajo los dos encabezamientos de valores iniciales llevamos los valores finales de la línea anterior, es decir, 1 y 4. Calculamos el valor de la variable MEDIO: (1+4)/2=2. En la columna "medio" escribimos 2 y en la columna Valor (medio) escribimos "B". Puesto que el valor de la letra "D" es mayor que el de la "B" modificaremos la variable PRIMERO. En la columna "primero final" escribimos 3 (MEDIO+1) y en la columna "último final" llevamos el valor anterior, o sea 4.

Ahora, la tabla de estado se presentará así:

Paso	Inicial		Medio	Valor (medio)	Final	
	Pr.	Ult.			Pr.	Ult.
1	1	21	11	M	1	10
2	1	10	5	E	1	4
3	1	4	2	B	3	4

Volvemos de nuevo al punto "A" y ya que el valor de la variable PRIMERO es menor que el de la variable ULTIMO, escribiremos la línea 4. En la columna "paso" escribimos 4, en la columna "primero inicial" escribimos 3 y en la columna "último inicial" escribimos 4. Calculamos el valor de la variable MEDIO: (3+4)/2=3. Escribimos 3 en la columna "medio" y C en la columna

"Valor (medio)". Puesto que la "D" es mayor que la "C", modificaremos solamente la variable "primero". Escribimos 4 en la columna de la variable "primero final" (MEDIO+1) y 4 en la columna "último final" (el valor anterior).

La situación de la tabla de estado quedará como sigue:

Paso	Inicial		Medio	Valor (medio)	Final	
	Pr.	Ult.			Pr.	Ult.
1	1	21	11	M	1	10
2	1	10	5	E	1	4
3	1	4	2	B	3	4
4	3	4	3	C	4	4

Vemos que el valor de la variable PRIMERO es igual al de la variable ULTIMO y, por ello, en la columna "paso" escribimos 5; a la columna "primero inicial" llevamos el valor 4 y a la columna "último inicial" llevamos también el valor 4. Calculamos el valor de la variable MEDIO ((4+4)/2) y escribimos el resultado en la columna "medio". En la columna "Valor (medio)" escribimos "D". Con ello hemos encontrado el elemento a buscar, que se encuentra en el cuarto lugar de la tabla.

Presentemos la situación final de la tabla de estado:

Paso	Inicial		Medio	Valor (medio)	Final	
	Pr.	Ult.			Pr.	Ult.
1	1	21	11	M	1	10
2	1	10	5	E	1	4
3	1	4	2	B	3	4
4	3	4	3	C	4	4
5	4	4	4	D		

Tratemos ahora de buscar la letra "J", es decir, un elemento que no existe en la tabla. Presentamos la tabla de estado tal como queda después del último paso, ya que damos por conocidos los razonamientos hechos en el caso de la búsqueda de la letra "D". Digamos solamente que en el paso 6 al preguntarnos si el valor de la variable PRIMERO es mayor que el de la variable ULTIMO y puesto que la respuesta es afirmativa, ello significa que el elemento no existe en la tabla.

Paso	Inicial		Medio	Valor (medio)	Final	
	Pr.	Ult.			Pr.	Ult.
1	1	21	11	M	1	10
2	1	10	5	E	6	10
3	6	10	8	H	9	10
4	9	10	9	I	10	10
5	10	10	10	L	10	9
6	10	9				

Acceso por el método Hash (al azar)

Este método de acceso ha adquirido tanta popularidad por ser el más rápido para acceso a un fichero mediante índice. Además, la tabla de claves ocupa menos espacio.

La figura 14 muestra en su apartado a) la situación después de la apertura del fichero. Existen las habituales 7 vías a nuestra disposición. La locomotora está preparada para aparcar el vagón "verde". Observamos que no se han utilizado todavía los compartimientos, sino que en todos ellos hemos introducido el valor -1. Para nosotros el valor -1 tiene el significado de compartimiento vacío o, lo que es lo mismo, de clave no existente.

En el acceso Hash se procede a aparcar los vagones (los registros) en el modo habitual: el primer vagón en la vía 0, el segundo en la vía 1 y así sucesivamente. Se observará que el vagón contiene un campo adicional, normalmente a "-1".

La originalidad del método Hash (comprobación al azar) consiste en transformar la clave, que suele ser de tipo alfabético, en un valor numérico. Las fórmulas para la transformación son numerosas, pero utilizaremos una bastante sencilla. Aun cuando las fórmulas o algoritmos sean complicadas puede suceder que a partir de claves diferentes se puede obtener el mismo número Hash. Este hecho se denomina "colisión" y se debe tener en cuenta.

El algoritmo utilizado es el siguiente: cada carácter que constituye la clave se transforma en su valor ordinal, o sea, en aquel valor binario mediante el cual está representado en la memoria del ordenador. Los valores binarios de los caracteres que ocupan posiciones distintas de la clave se sumarán en el totalizador "X", mientras que los otros valores se sumarán en "Y". A continuación, los resultados de "X" e "Y" se dividen por 256 y los restos de las divisiones correspondientes se llevan a las variables R1 y R2, respectivamente. Se multiplica R2 por 256 y se le suma R1; el resultado se lleva a "Z". Finalmente, dividimos "Z" por 7 y almacenamos

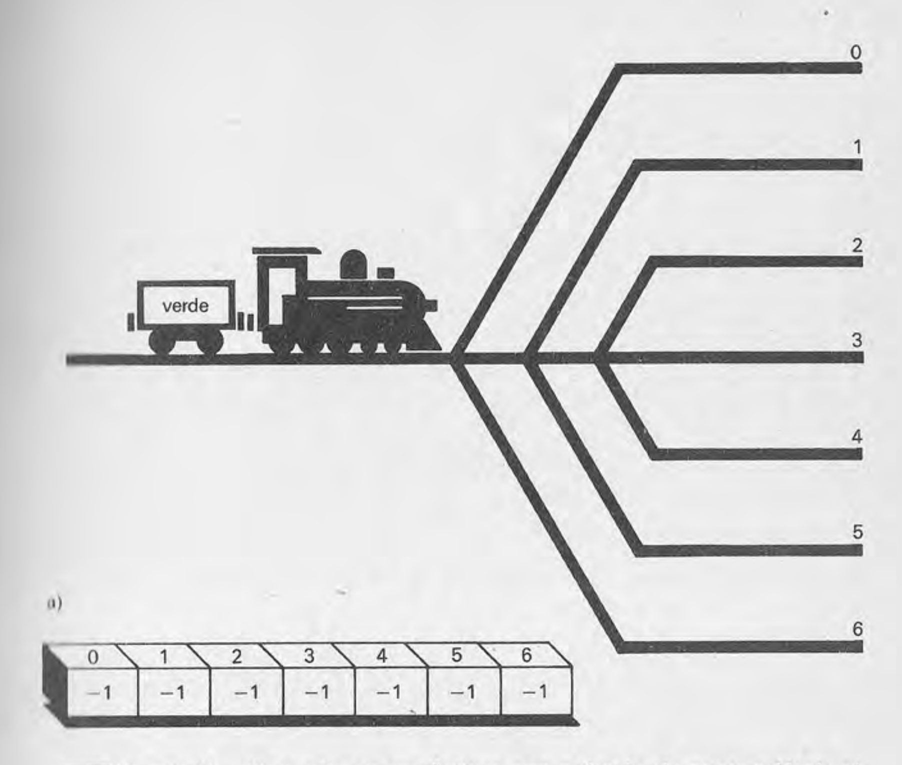


Figura 14.—Creación de un fichero con método de acceso Hash. a) Situación antes de procesar los datos correspondientes a "verde".

el resto en "H". "H" será nuestro número Hash. Como divisor de "Z" se utiliza un número primo, teóricamente el más próximo al número de registros que constituyen el fichero. En la práctica suele ser un número inferior, con el fin de no desperdiciar memoria.

En la siguiente tabla se resume la transformación de nuestras 6 claves en su número Hash.

Clave	X	Y	R1	R2	Z	H
Verde	337	201	81	201	51537	3
Rojo	334	226	78	226	57934	2
Añil	197	213	197	213	54725	6
Dorado	311	209	55	209	53559	2
Gualdo	308	318	52	62	15924	6
Violeta	206	207	206	207	52972	3

Se observan muchas "colisiones". Utilizando como divisor 17 no habríamos obtenido ninguna colisión, pero la tabla estaría constituida por 17 elementos, aunque los registros sean solamente 6.

En la parte b) de la figura 14 se muestra la situación después de la grabación de registro "verde". El número Hash de "verde" es 3; por ello abrimos el compartimiento número 3 y observamos que contiene el valor -1, lo que significa que está vacío. Por consiguiente, introducimos el valor 0 (número de la vía libre). En el vagón "verde" añadimos el valor -1 (veremos luego para qué sirve) y luego lo aparcamos en la vía 0.

La siguiente vía libre es la número 1. En el apartado a) de la figura 15 se muestra la situación después de la grabación del registro "rojo". El número Hash de "rojo" es 2; abrimos el com-

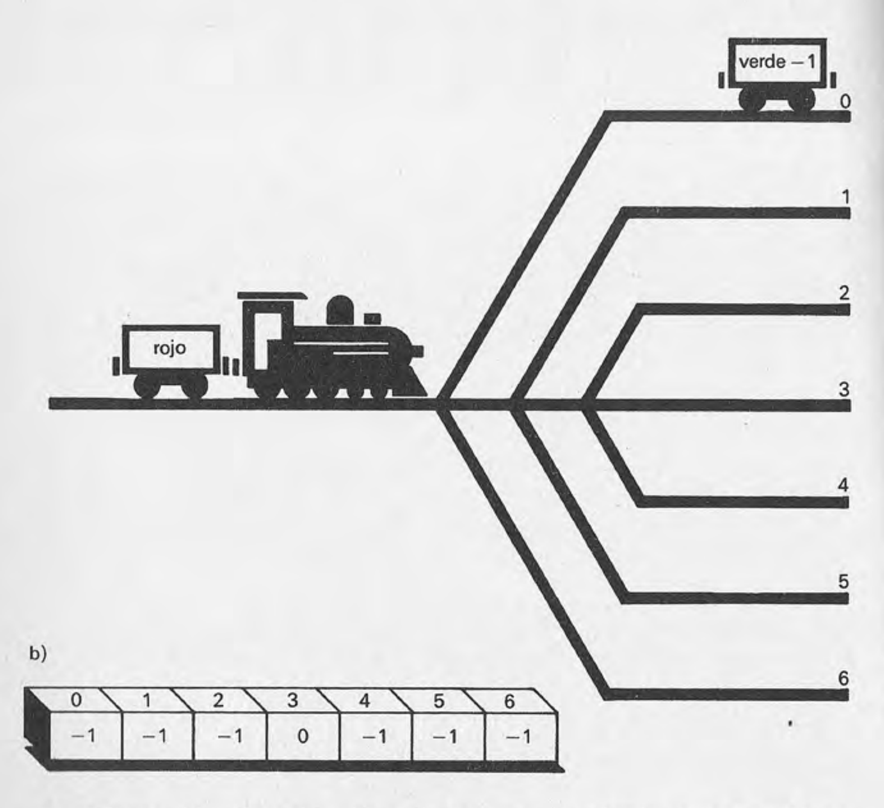


Figura 14.—Creación de un fichero con el método de acceso Hash. b) Situación después de la escritura de "verde".

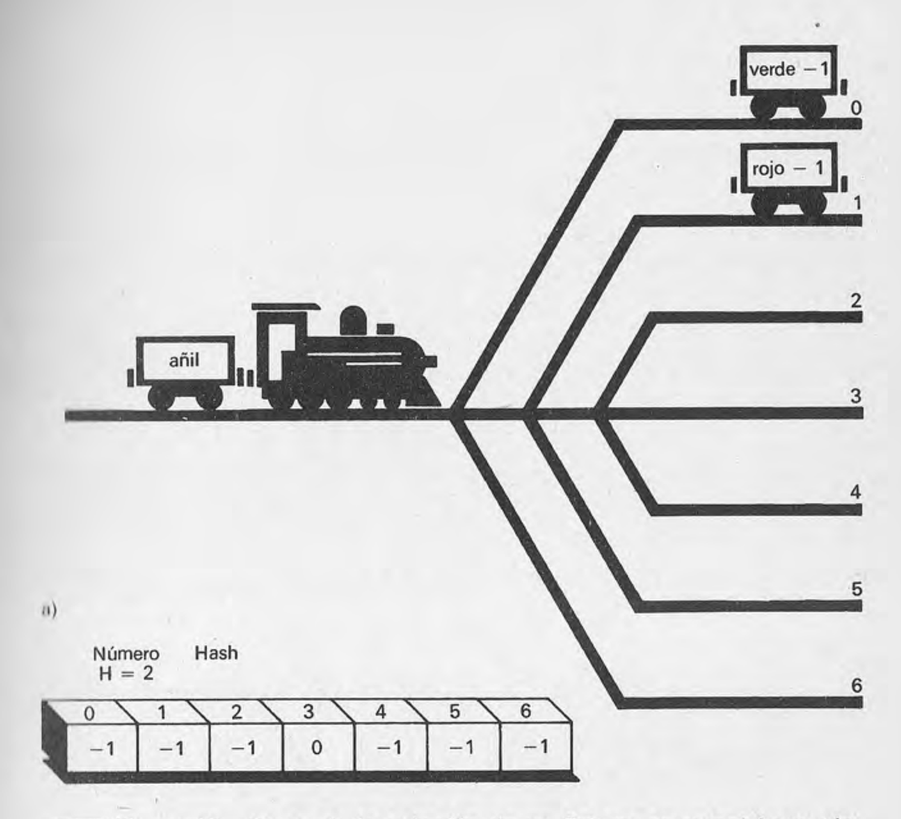


Figura 15.—Pasos de la crèación de un fichero secuencial con técnica de acceso Hash. a) La situación después de la grabación de "rojo".

partimiento número 2 y observamos que contiene el valor -1, es decir, está vacío. Ahora, introducimos en dicho compartimiento el valor 1, que es el número de la vía libre. En el vagón "rojo" añadimos el valor -1 y aparcamos el vagón en la vía 1.

La siguiente vía libre es la número 2. Es fácil comprender cuál será la situación después de la grabación del registro "añil": el número Hash de añil es 6, así que abrimos el compartimiento número 6 y observamos que está vacío (contiene -1); introducimos el valor 2, que es el número de la vía en donde aparcamos el vagón "añil". En este último vagón añadimos el valor -1 y lo depositamos en la vía número 2. La siguiente vía libre será la número 3. Recomendamos para los pasos considerados y los que veremos más adelante que se tenga a la vista la tabla de generación de los diversos números Hash.

En el apartado b) de la figura 15 se muestra la situación después de la grabación del registro "dorado". El número Hash de "dorado" es 2; al abrir el compartimiento 2 observamos que está ocupado ya: nos encontramos ante la primera "colisión". Para subsanar este inconveniente, escribimos en el vagón "dorado" el valor "1", es decir, el valor contenido hasta ahora en el compartimiento y luego aparcamos el vagón en la vía 3. La siguiente vía libre es la número 4. El compartimiento número 2 ha quedado, por así decirlo, con la puerta abierta de par en par; el valor "1" se ha extraído y ahora hemos introducido el valor "3", que corresponde al número de la vía en dónde ha estado aparcado el vagón "dorado". También en la grabación del registro "dorado" se tiene otra colisión. Añadimos al vagón "gualdo" el valor 2, extraído del compartimiento 6 y luego aparcamos el vagón en la vía 4. La siguiente vía libre será la número 5. También el compartimiento número 6 ha quedado abierto y hemos puesto en su interior el valor 4, o sea, el valor de la vía en donde se ha depositado el valor "gualdo".

En el apartado c) de la figura 15 se muestra la situación después de la grabación del registro "violeta", cuyo número Hash es 3 y que, por consiguiente, produce otra colisión. Al vagón "violeta" le añadimos el valor "0" contenido en el compartimiento y ponemos el vagón en la vía 5, mientras que en el compartimiento 3 introducimos el valor 5. La siguiente vía libre es la número 6.

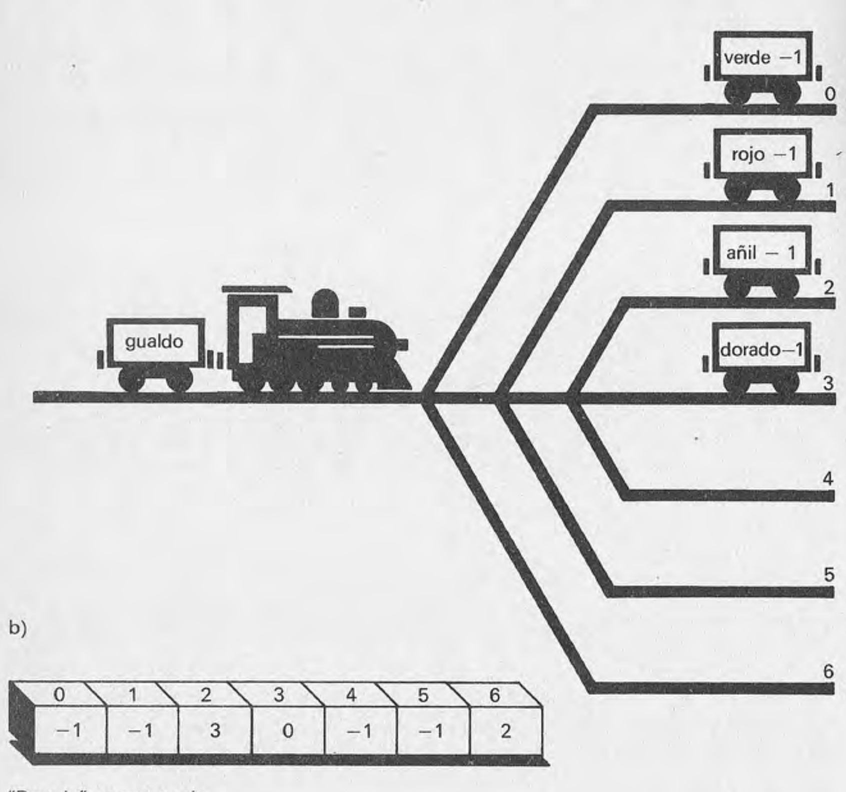


Figura 15.—Pasos de la creación de un fichero secuencial con técnica de acceso Has. b) Se tiene con "dorado" la primera colisión; el compartimiento 2 contendrá el número de la vía "dorado", cuyo registro "apunta" al interior (rojo);

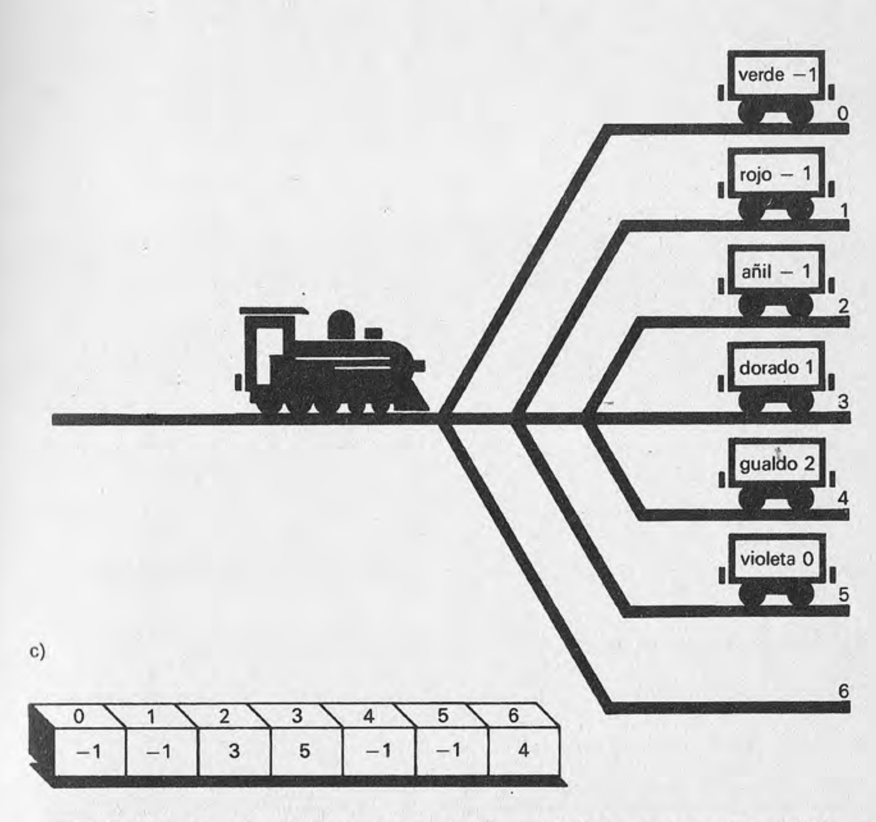


Figura 15.—Pasos de la creación de un fichero secuencial con técnica de acceso Hash. c) Situación después de la escritura del último registro.

En la figura 16 se muestra la situación final de las vías, de los compartimientos y la composición del tren... perdón, del fichero. El tren tiene uno o más vagones con las claves; en circunstancias normales, dichos vagones se colocan inmediatamente después de la locomotora.

Pasemos ahora al examen de las modalidades de búsqueda con el empleo del método Hash.

En el apartado a) de la figura 17 se muestra la búsqueda de la clave "dorado". El número Hash correspondiente a esta clave

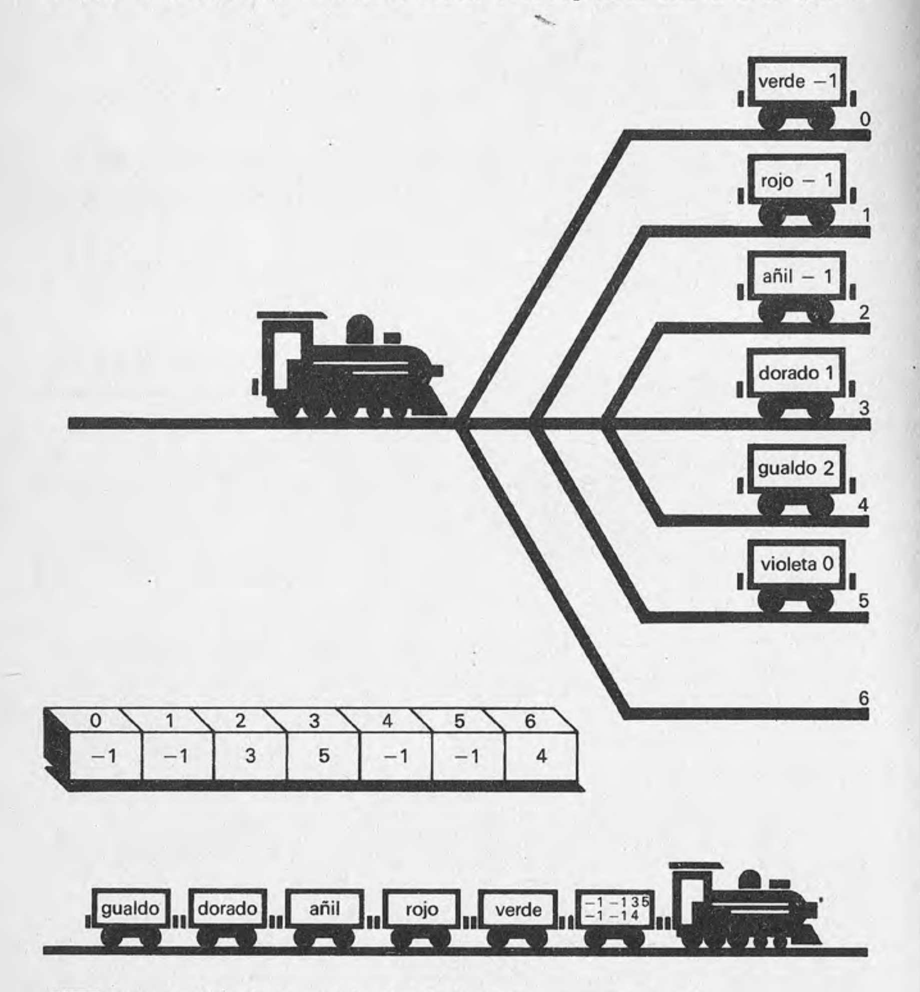


Figura 16.—Disposición del fichero por acceso Hash.

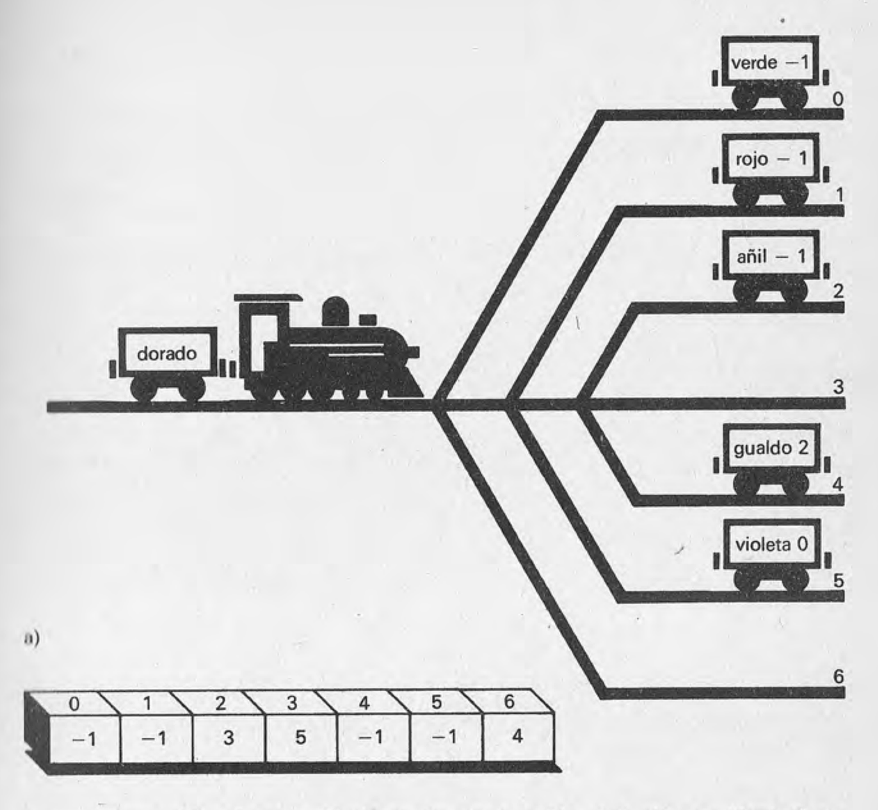


Figura 17.—Acceso Hash. a) La generación del número Hash tiene efecto inmediato.

es 2 y, en consecuencia, abrimos el compartimiento 2, que contiene el valor 3. La locomotora se pone en marcha hacia la vía número 3, en donde encuentra el vagón "dorado" de forma inmediata y sin problemas.

Supongamos ahora que tratamos de buscar el vagón "verde", cuyo número Hash es 3. Abrimos el compartimiento 3 y nos encontramos con que contiene el valor 5. La locomotora se llevará a la vía 5; la situación se ilustra en el apartado b) de la figura 17. En la vía 5 nos encontramos, en realidad, el vagón "violeta". Observamos, no obstante, que éste contiene el valor 0 y no -1, lo que significa que hay una colisión y que en la vía 0 podemos encontrar el vagón "verde". Realizamos otro desplazamiento con la locomotora, esta vez a la vía 0, y allí encontramos, efectivamente, el vagón "verde".

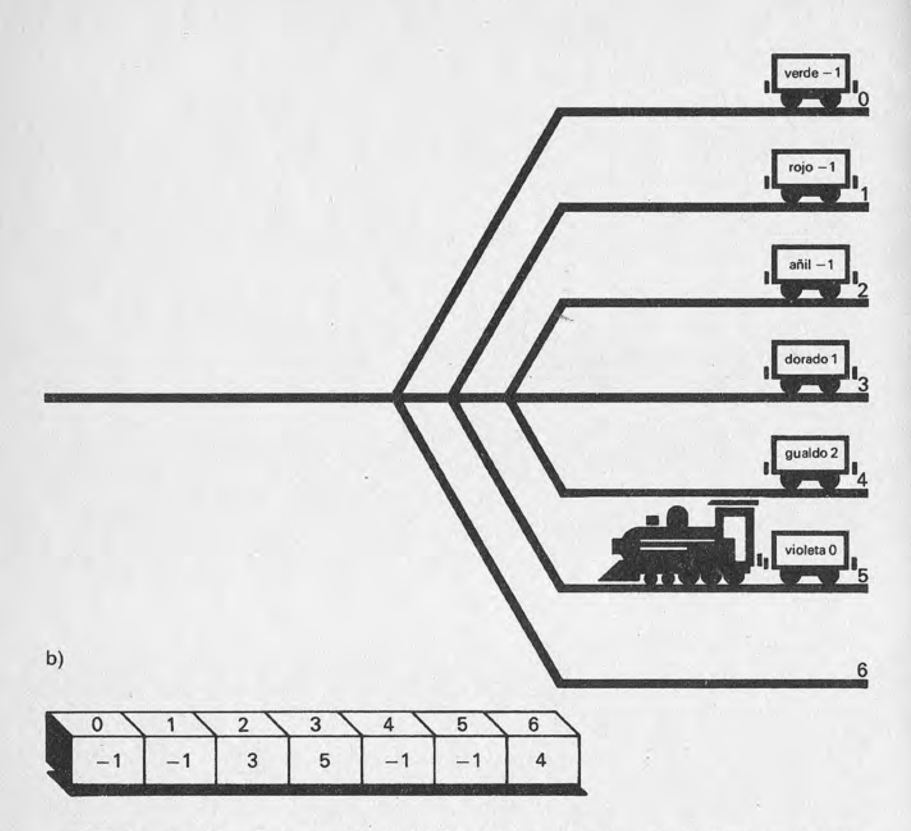


Figura 17.—Acceso Hash. b) En el caso de "verde", el número contenido en el compartimiento se lleva a la vía en la que se encuentra en realidad "violeta".

Supongamos ahora que hay que buscar el vagón "negro". Naturalmente, sabemos que no existe, pero el ordenador no es tan intuitivo como una persona. El número Hash de "negro" sería 5 (X=224, Y=206, R1=224, R2=206, Z=52960). Abrimos el compartimiento número 5 y nos encontramos con que contiene el valor -1, lo que significa que está vacío, o sea, que no existe dicha clave.

Busquemos ahora la clave "rosa", que también sabemos que no existe. El número Hash de "rosa" es 6 (X=229, Y=212, R1=220, R2=212, Z=54501). Abrimos el compartimiento 6 y encontramos el valor 4; con la locomotora nos desplazamos a la vía 4 y tendremos la situación que se ilustra en la figura 18 a). Observamos que en la vía 4 está el vagón "gualdo" y que no es precisamente el buscado; pero dicho vagón lleva el valor 2, indicando una colisión.

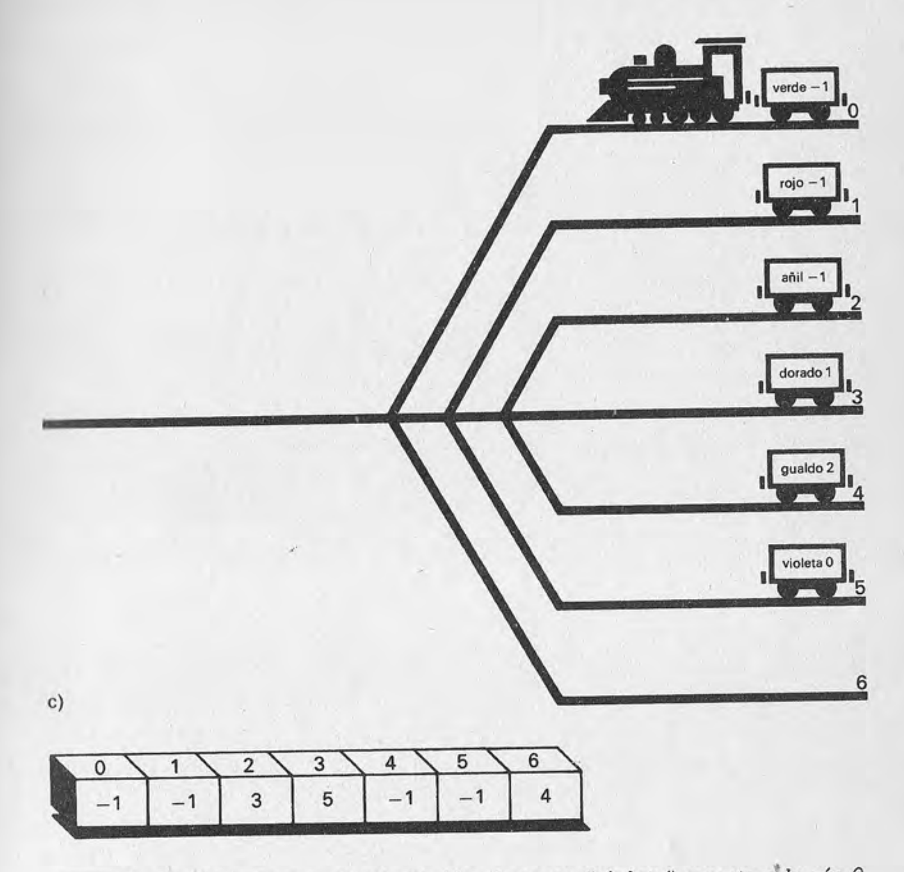


Figura 17.—Acceso Hash. c) No obstante, "violeta" apunta a la vía 0, y en otro desplazamiento del tren se encuentra el "verde" que se buscaba.

Con la locomotora nos desplazamos a la vía 2, en donde encontramos el vagón "añil" (Fig. 18 b). Tampoco este vagón es el buscado; en él encontramos el valor "-1", lo que quiere decir que no hay otras colisiones o, dicho de otro modo, que la clave "rosa" no existe en el archivo.

Veamos ahora cómo actuar para borrar un registro. Para hacerlo, es preciso modificar los punteros. Se suele indicar en el propio registro que se ha borrado utilizando un campo adecuado. La figura 19 a) muestra la situación después de borrar el registro "gualdo". El número Hash de "gualdo" es 6. Abrimos el compartimiento número 6 y nos encontramos con el valor 4. Pasamos a la

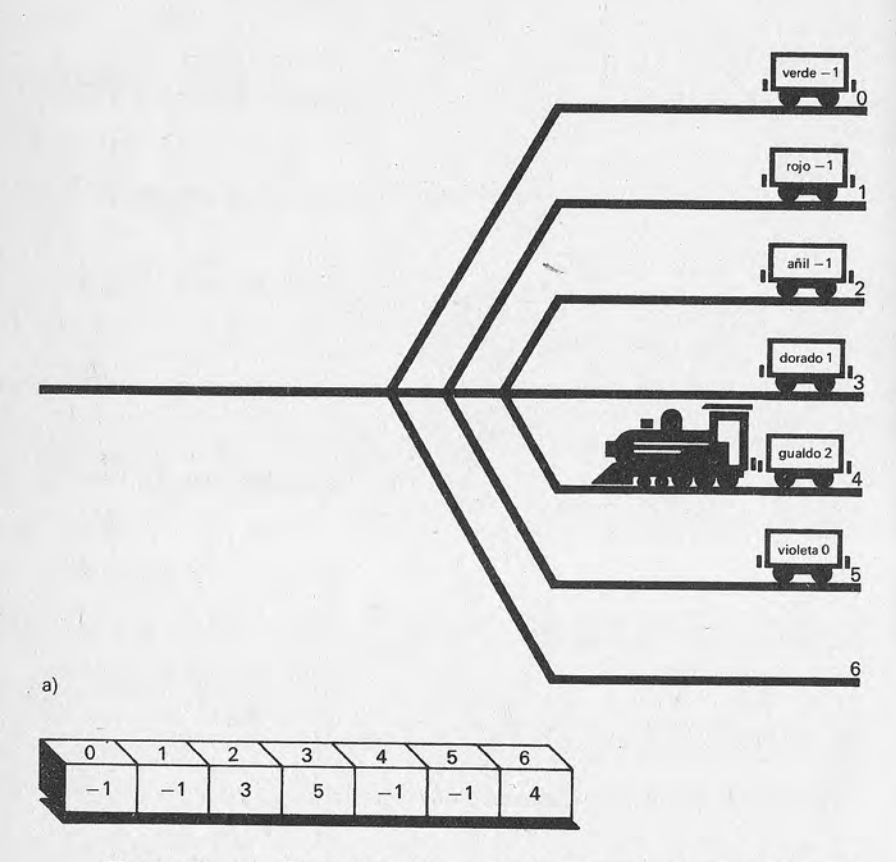


Figura 18.—Dos fases de la búsqueda de claves inexistentes ("rosa") con colisiones. a) Primera fase de desplazamiento.

vía 4, encontramos el vagón "gualdo" y observamos que su puntero es 2; este valor lo introducimos en el compartimiento 6, sustituyendo al valor existente con anterioridad. Además, indicamos en el vagón que la información se ha anulado. En el apartado b) de la misma figura se muestra la situación después de la anulación del registro "añil".

Las ventajas del acceso Hash son notorias: en la generación de un fichero de acceso Hash, los registros nunca sufren desplazamiento alguno. El conjunto de las claves ocupa poca memoria. Si las colisiones son pocas, el acceso a cualquier registro es muy rápido.

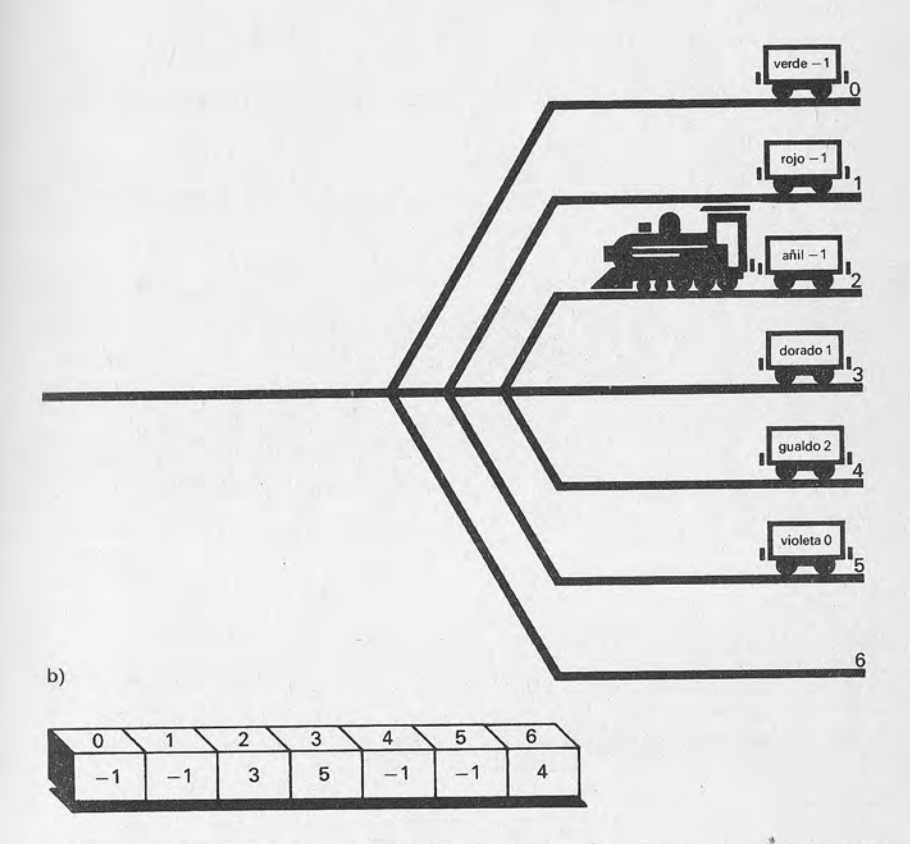


Figura 18.—Dos fases de la búsqueda de claves inexistentes ("rosa") con colisiones. b) Puesto que el nuevo registro no es "rosa" y "añil" tampoco apunta a ningún otro, se llega a la conclusión de que la clave buscada no está en el archivo.

No obstante, en los registros con acceso Hash existen también dos inconvenientes importantes:

- no se puede leer el fichero en orden creciente o decreciente de clave si no se conocen las claves;
- no es posible realizar una búsqueda en clave parcial. Si utilizamos como clave el apellido no es posible, por ejemplo, buscar todos los apellidos que comienzan por "ro".

También para el acceso Hash es válido el conocido proverbio de que "no se puede querer el beneficio sin cargar con el gas-

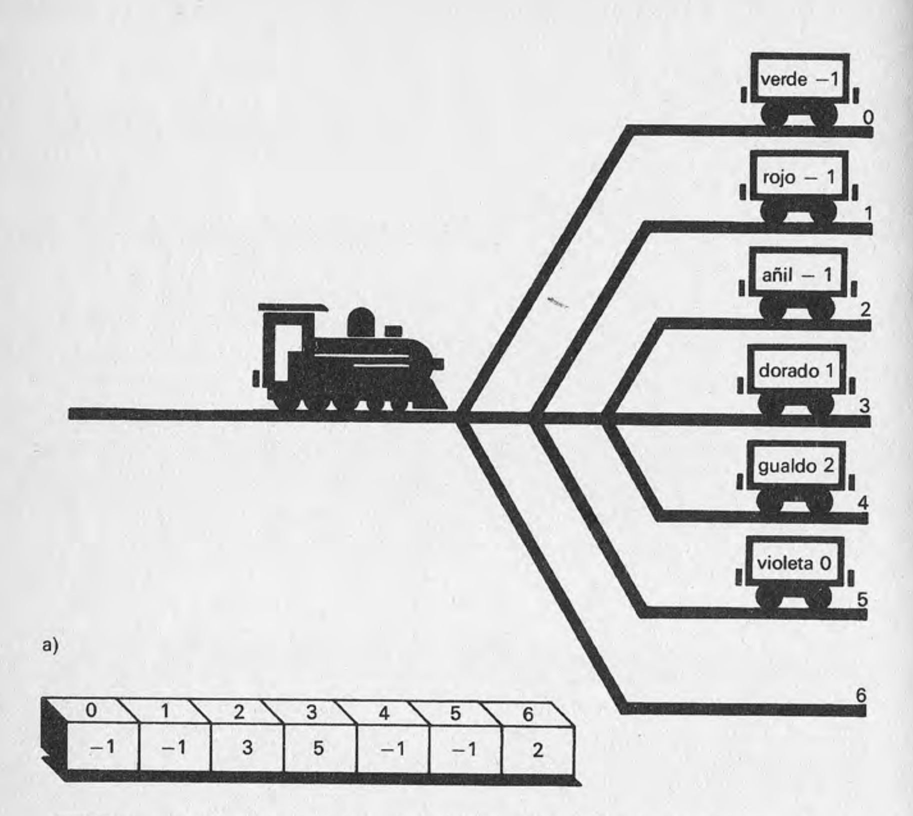


Figura 19.—Situación después de la anulación de dos registros "gualdo" en a) y "añil" en b). Observe la sustitución de los punteros.

to". Con frecuencia, el acceso Hash es muy válido para búsqueda en archivos de tipo "vocabulario" o "enciclopedia", en donde se sabe exactamente qué clave buscar.

El árbol binario

Presentamos uno de los más sencillos ejemplos de árbol binario. Utilizaremos también los habituales 6 registros y el simbolismo del tren y de los vagones, aunque las vías estarán dispuestas de una manera diferente.

Al comienzo se parte con un nudo ferroviario, la "raíz" del árbol, que contiene una vía muerta en la que se aparcará el vagón, inmediatamente antes de que haya un cambio que permita tener acceso a la vía de la izquierda o a la vía de la derecha. En con-

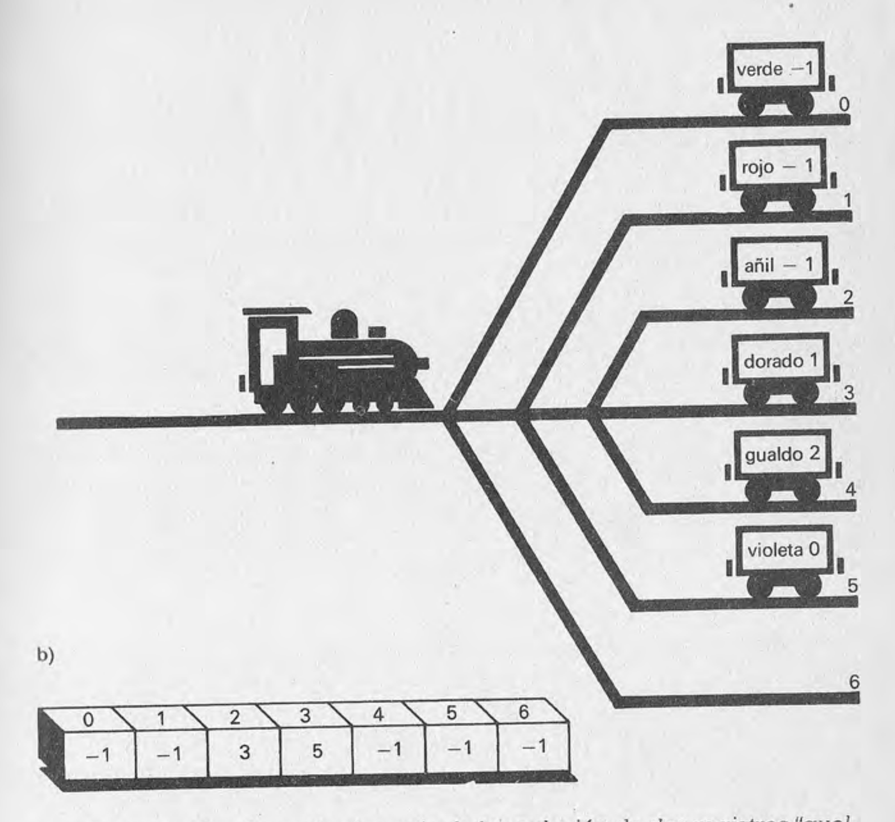


Figura 19.—Situación después de la anulación de dos registros "gualdo" en a) y "añil" en b). Observe la sustitución de los punteros.

diciones normales las dos vías estarán cerradas. Al colocar un nuevo vagón éste se enviará a la vía de la izquierda si su clave es menor que la clave del vagón existente en la vía muerta y, de no ser a sí, se destinará a la vía de la derecha. Por izquierda y derecha entendemos nuestra izquierda o derecha y no la que "ve" el conductor de la locomotora.

Recordemos, de forma muy breve, cuanto se dijo en otros volúmenes de nuestra biblioteca (por ejemplo, en los números 8 y 9), acerca de que un árbol binario es una estructura de datos que, partiendo de un nudo raíz, puede proliferar en diversas parejas de "nudos hijos". En cada uno de los nudos componentes se pueden engendrar otros dos nudos hijos (lo que da lugar a una subdivisión en árboles secundarios que tienen como raíz el nudo anterior). Cada nudo se numera a partir del 0 en adelante y contiene dos punteros. El de la izquierda, si es diferente de 0, indica

el siguiente número que se encontrará siguiendo la vía de la izquierda, y el puntero de la derecha, si contiene un valor diferente de 0, puede significar dos cosas: si el valor es positivo indica el número del nudo que se encuentra siguiendo la vía de la derecha, y si es negativo, su valor absoluto indica cuál es su raíz.

En el apartado a) de la figura 20 se muestra la situación inicial inmediatamente después de la apertura del fichero y con el vagón "verde" dirigiéndose hacia una vía. En el apartado b) de la misma figura se muestra el vagón "verde" colocado en su vía muerta; el nudo ferroviario que se ha creado presenta dos vías, ambas cerradas (existe en ellas la palabra "NO").

En la parte c) se tiene la situación después de la grabación del registro "rojo". El vagón rojo se ha encaminado a la vía de la

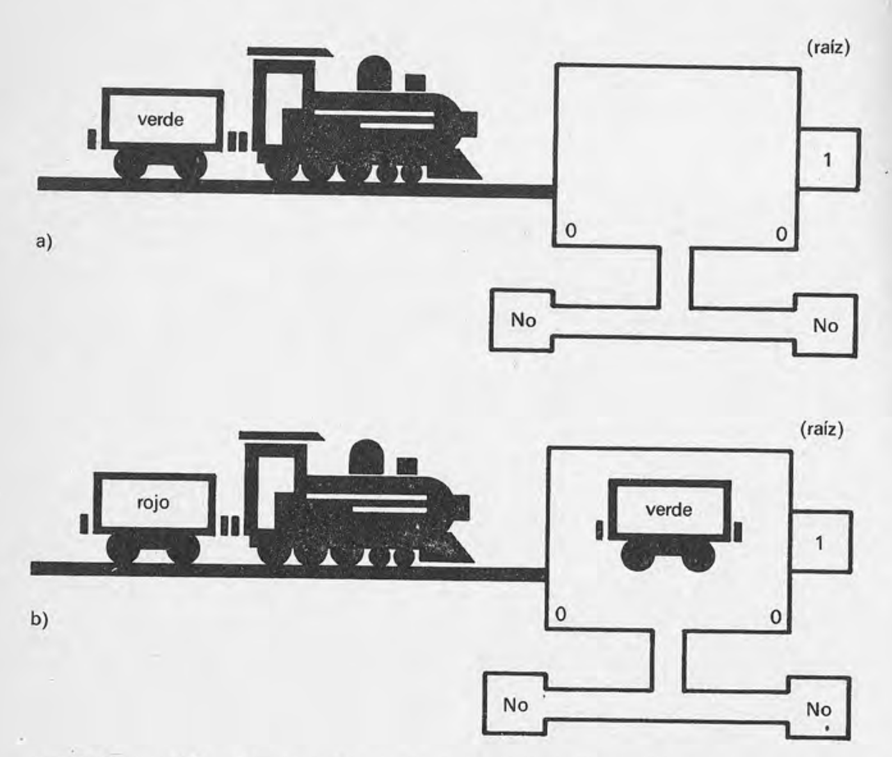


Figura 20.—Diversas fases de la creación de un fichero organizado en árbol binario. En a) y b) la situación antes y después de la inserción de "verde". En c) y d) se indica cómo queda después de la inserción de "rojo" y "añil". Se observan los punteros negativos que apuntan a los registros situados más arriba.

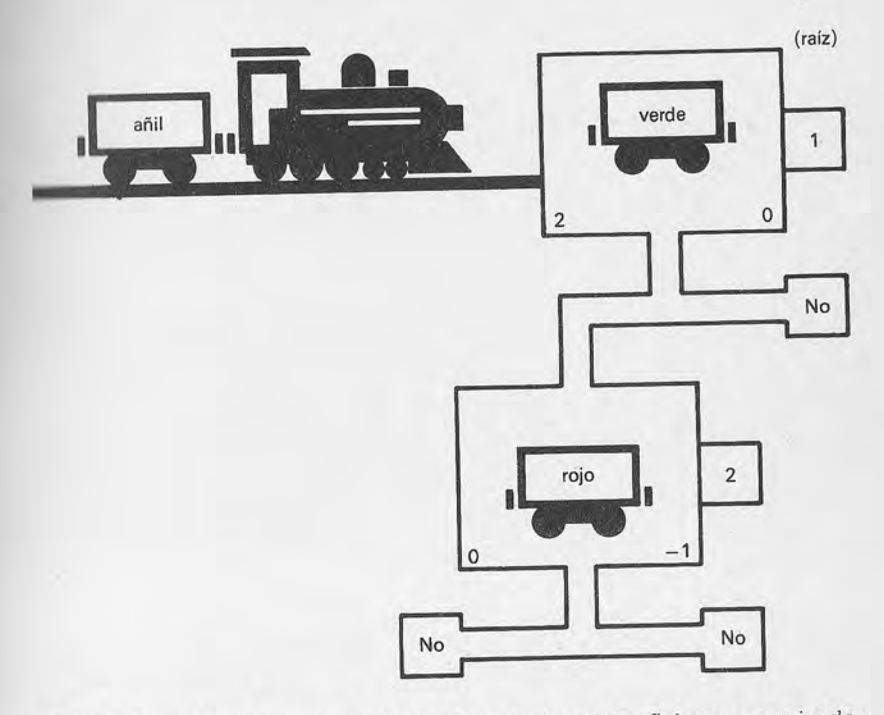


Figura 20.—Diversas fases de la creación de un fichero organizado en árbol binario. En a) y b) la situación antes y después de la inserción de "verde". En c) y d) se indica cómo queda después de la inserción de "rojo" y "añil". Se observan los punteros negativos que apuntan a los registros situados más arriba.

izquierda del nudo "verde", porque la clave de "rojo" es menor que la clave "verde". Las dos vías de rojo están cerradas y el puntero izquierdo de "verde" contendrá ahora el valor 2. El puntero izquierdo de "rojo" contiene 0, mientras que el de la derecha contiene "–1" (la clave "verde" sigue a la de "rojo"). De forma análoga en d) se muestra la situación después de la grabación de "añil". La locomotora parte de arriba, encuentra el nudo ferroviario "verde" y, habida cuenta de que la clave de "añil" es menor que la de "verde", se dirige a la vía de la izquierda. Luego encuentra el nudo "rojo" y al ser la clave de "añil" menor que la de "rojo" se dirige a la vía de la izquierda y crea un nuevo nudo con dos vías que, por ahora, están cerradas. En la vía muerta deposita el vagón "añil". El puntero izquierdo de "rojo" contiene ahora 3. El puntero izquierdo de "añil" contiene 0 y el de la derecha "–2" (antes de "añil" está "rojo").

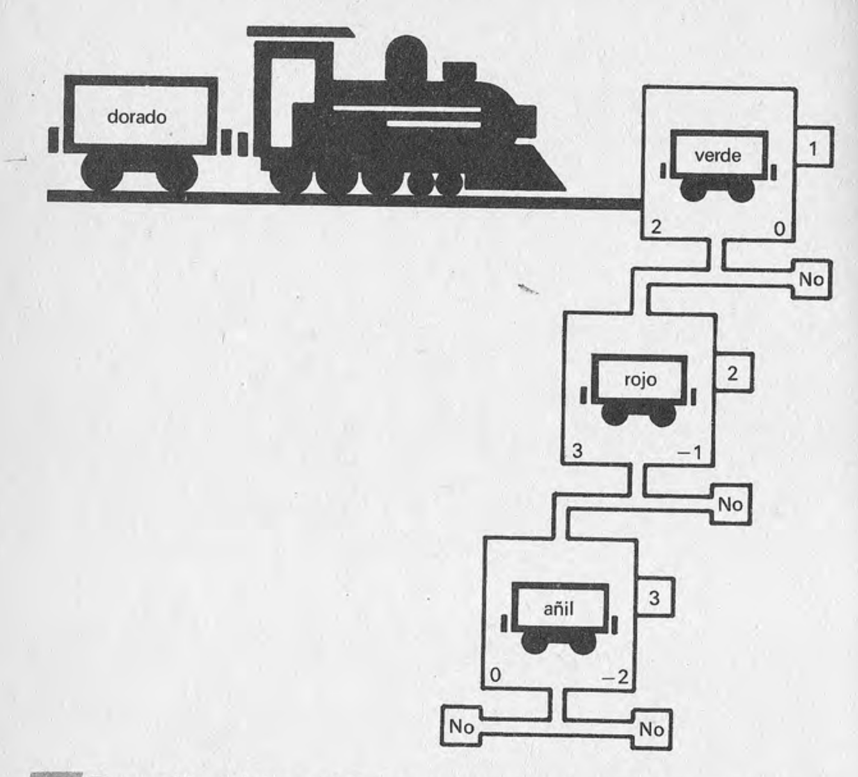


Figura 20.—Diversas fases de la creación de un fichero organizado en árbol binario. En a) y b) la situación antes y después de la inserción de "verde". En c) y d) se indica cómo queda después de la inserción de "rojo" y "añil". Se observan los punteros negativos que apuntan a los registros situados más arriba.

En la figura 21 encontramos la situación después de la inserción del último registro. Con un mínimo de imaginación no resulta difícil reconstruir las etapas intermedias, que aquí describimos "de corrido". Para la inserción del registro "dorado", la locomotora vuelve a partir desde arriba, encuentra el nudo "verde" y se dirige a la vía de la izquierda, encuentra luego el nudo "rojo" y toma de nuevo la vía de la izquierda. La locomotora encuentra luego el nudo "añil", pero esta vez se encamina a la vía de la derecha, creando el nudo ferroviario "dorado" con dos vías cerradas por ahora. El vagón "dorado" se deposita en la vía muerta del nudo "dorado". El puntero derecho de "añil" contiene el valor 4 y su valor anterior se ha transferido al puntero derecho de "dorado". En el puntero izquierdo de "dorado" se pone el valor 0.

Figura 21.—Situación del árbol binario después de la inserción del último registro.

Por lo que respecta a la inserción del registro "gualdo", la locomotora, después de haber encontrado el nudo "verde", gira a la izquierda y luego encuentra el nudo "rojo" y se dirige a la vía de la izquierda, en donde encuentra el nudo "añil" y desemboca en la vía de la derecha; finalmente encuentra el nudo "dorado", toma la vía de la derecha y crea un nuevo nudo ferroviario en donde deposita el vagón "gualdo" y crea dos nuevas vías, cerradas por ahora. El puntero derecho de "dorado" contiene ahora el valor 5 y su valor anterior se ha colocado en el puntero derecho del registro "gualdo". El puntero izquierdo de "gualdo" contiene 0. En este punto, dejamos como ejercicio para el lector la no difícil inserción del registro "violeta".

Cómo se efectúa la búsqueda en un árbol binario

Partamos de la situación representada en la figura 21. Supongamos que queremos buscar la clave "dorado". Con la locomotora en el nudo raíz nos encontraremos con la clave "verde"; nos dirigiremos entonces a la vía de la izquierda porque la clave de búsqueda es menor que la del nudo. La locomotora encuentra sucesivamente el nudo "rojo", se dirige a la vía de la izquierda, en donde encuentra el nudo "añil", y sigue aquí por la vía de la derecha, en donde encuentra el nudo "dorado".

Por el contrario, si queremos buscar la clave "rosa", que sabemos que no existe, la locomotora, después de haber encontrado el vagón "verde" se dirigirá a la vía de la izquierda, y luego encontrará el nudo "rojo". Entonces debería pasar a la vía de la derecha ("rosa" > "rojo"), pero no puede proseguir porque la vía está cerrada. Ello significa que el vagón "rosa" no existe.

En la figura 22 se muestra la situación después de la eliminación de la clave "dorado"; es oportuno observar cómo se ha realizado la anulación del nudo, cambiando adecuadamente los punteros (la clave "dorado", aunque físicamente anulada, sigue existiendo).

El árbol binario es una estructura válida sobre todo porque permite la actualización, la creación y el borrado manteniendo la ordenación de los datos. Para que las operaciones de búsqueda sean eficientes es preciso, no obstante, que el árbol esté "equilibrado".

Un árbol se dice que está desequilibrado si la mayoría de las vías, bien sean de la derecha, bien de la izquierda, están cerradas. Por ejemplo, un árbol creado a partir de claves ya ordenadas (tanto en sentido ascendente como descendente) está desequilibrado sin duda alguna. Las técnicas especiales que permiten crear y mantener árboles equilibrados quedan fuera del objeto de esta monografía. El árbol binario puede ser recorrido de modo que se lean todas las claves en orden y es posible realizar búsquedas con claves parciales.

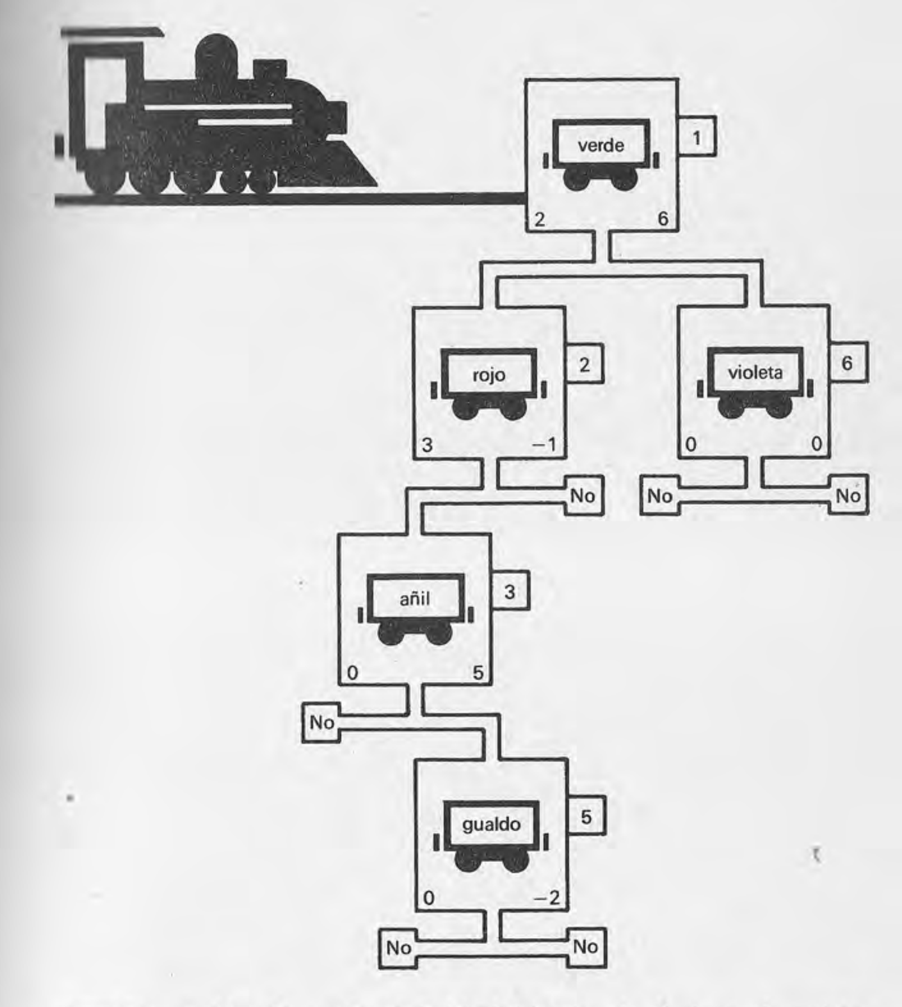


Figura 22.—Situación del árbol binario al eliminar el registro "dorado".

Vamos a repetir ahora algunos términos técnicos que se suelen utilizar en el estudio de las estructuras en árbol. Un árbol se suele representar con la raíz arriba y con un crecimiento desde arriba hacia abajo, denominándose "nudo" el punto en donde se

bifurcan las "ramas" y "hojas" a los puntos en donde no hay bifurcaciones, sino un registro. En cada nudo y cada hoja hay un registro. Cada nudo puede tener uno o dos hijos, denominados también "subnudos" o nudos secundarios; el hijo de la izquierda debe tener un valor inferior al valor del padre. Las hojas no tienen hijos y, en tal caso, el puntero contiene el valor "NIL" (que hemos representado por NO en las figuras).

Existen varias maneras de recorrer un árbol. La que se utiliza en condiciones normales se denomina "en orden", porque sigue las claves de manera ordenada. La regla aplicable al recorrido en orden es la siguiente: pasar primero a la hoja de la izquierda, luego por su nudo y, finalmente, por la hoja de la derecha.

La figura 23 muestra el diagrama de flujo de la impresión de un árbol en orden. Para comprender mejor el procedimiento utilizaremos la tabla de los estados que, como se dijo anteriormente, proporciona el contenido de algunas variables en las diversas fases. Presentamos primero la situación final de la tabla y luego la situación paso a paso. He aquí la situación final:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—
2	2	3	-1	3	—	—
3	3	0	4	4	—	añil
4	4	0	5	5	—	dorado
5	5	0	-2	2	—	gualdo
6	2	3	-1	1	sí	rojo
7	1	2	6	6	sí	verde
8	6	0	0	0	—	violeta

La tabla se desarrolla en 8 pasos y está constituida por varias columnas.

La columna "Paso" está numerada de forma progresiva e indica el número de la fase que se está desarrollando.

La columna "X Primero" indica el valor de la variable numérica "X" al comienzo de fase.

La columna "Izq(x)" indica el contenido del puntero de la izquierda existente en el nudo "X".

La columna "Der(x)" indica el contenido del puntero de la derecha existente en el nudo "X".

La columna "X Después" indica el contenido de "X" al final de la fase, pues en cada fase la variable "X" cambia de valor.

La columna "Entrada en B" indica si el reenvío al algoritmo se hizo al punto "B" y no al punto "A".

La columna "Impresión" indica qué se imprime; el símbolo "–" indica que no se imprime nada en absoluto.

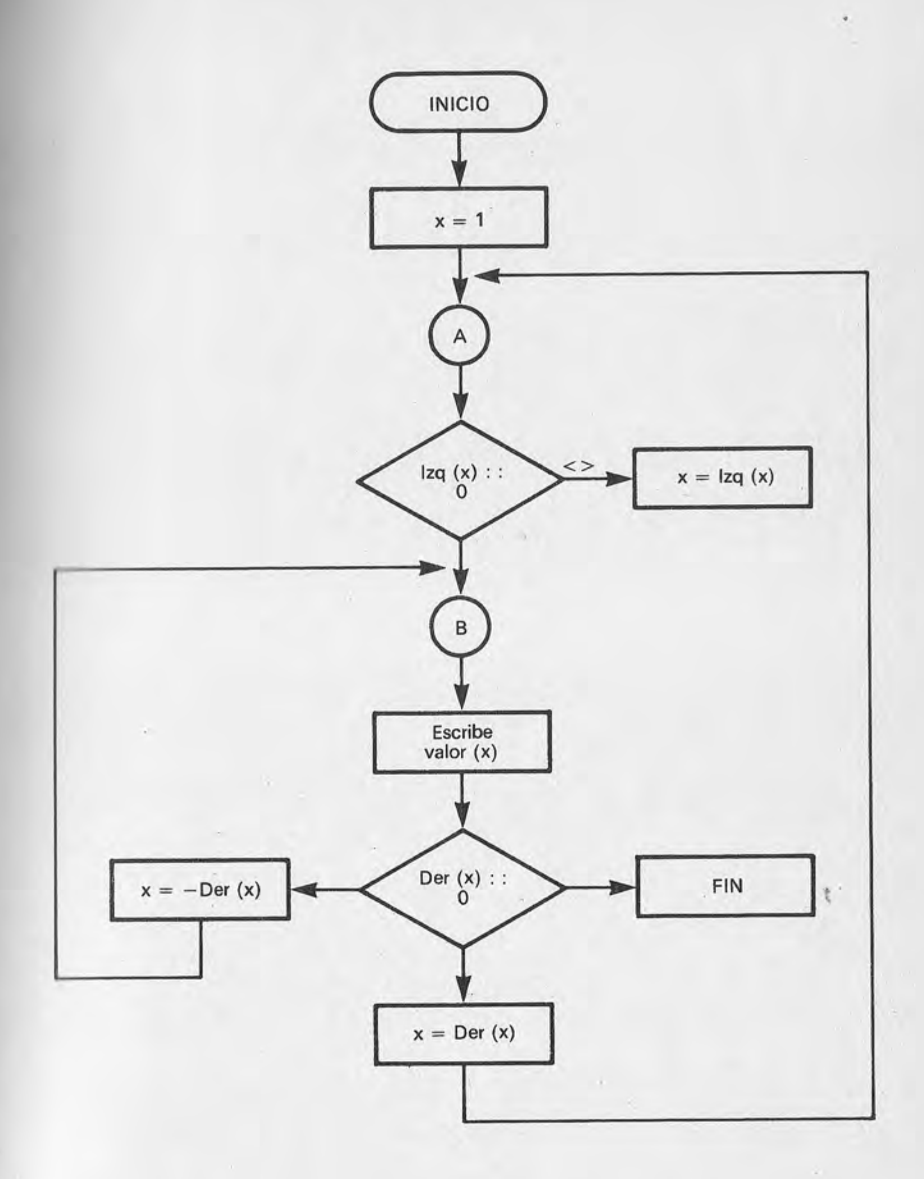


Figura 23.—Gestión de un fichero en árbol binario. Diagrama de flujo de la técnica de recorrido "en orden" del árbol. OBSERVACION: A y B identifican los puntos de nueva entrada y se podrían eliminar (sirven solamente para la descripción de los diversos pasos en la "tabla de los estados").

Veamos paso a paso lo que sucede. La situación del primer paso es:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—

Al comienzo del primer paso la variable X contiene el valor 1; el puntero de la izquierda contiene el valor 2 y el de la derecha contiene el valor 6. Esto significa que al "doblar la esquina" a la izquierda se encuentra el nudo 2, que debe tener un valor inferior al nudo 1, y al girar a la derecha se encuentra el nudo 6, que debe tener un valor superior.

Para recorrer el árbol "en orden" tendremos que examinar primero todas las vías de la izquierda, y para ello obligamos a que X tenga el valor contenido en el puntero izquierdo. Naturalmente, no se realiza ninguna impresión y la situación después del paso 2 será la siguiente:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—
2	2	3	-1	3	—	—

Estamos en el segundo nudo: el puntero izquierdo contiene el valor 3, mientras que el puntero derecho contiene el valor "-1". Obligamos a que en "X" exista el contenido del puntero izquierdo; es decir, 3. Naturalmente, no se realiza ninguna impresión.

La situación después del paso 3 es:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—
2	2	3	-1	3	—	—
3	3	0	4	4	—	añil

Estamos en el tercer nudo: el puntero izquierdo contiene 0 y el puntero derecho contiene el valor 4. Hemos llegado a una hoja y por ello podemos imprimir el contenido del registro. Puesto que el puntero derecho no contiene un valor negativo, introducimos

en "X" el valor 4 y volvemos al punto "A" (hacer siempre referencia al diagrama de flujo de la figura 23).

La situación después del paso 4 será la siguiente:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—
2	2	3	-1	3	—	—
3	3	0	4	4	—	añil
4	4	0	5	5	—	dorado

Nos encontramos en el cuarto nudo: el puntero izquierdo contiene 0 y el puntero derecho 5.

En primer lugar, imprimimos el contenido del registro (dorado) y luego introducimos en "X" el contenido del puntero derecho, porque este valor es positivo. Luego volvemos al punto "A".

La situación después del paso número 5 será:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	1	6	2	—	—
2	2	3	-1	3	—	—
3	3	0	4	4	—	añil
4	4	0	5	5	—	dorado
5	5	0	-2	2	—	gualdo

Nos encontramos en el nudo 5 y puesto que el puntero izquierdo contiene 0, imprimimos el contenido del registro. El puntero derecho contiene un valor negativo (-2) y lo transferimos con signo positivo a "X". A continuación, volvemos a entrar en la subrutina, pero esta vez en el punto B.

La situación después del paso 6 es la siguiente:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—
2	2	3	-1	3	—	—
3	3	0	4	4	—	añil
4	4	0	5	5	—	dorado
5	5	0	-2	2	—	gualdo
6	2	3	-1	1	sí	rojo

Puesto que "X" contiene el valor 2, imprimiremos el contenido del nudo número 2 (es decir, "rojo"). Luego observamos que el puntero derecho es negativo y lo transferimos a la variable X con el signo positivo. Volvemos a entrar en la subrutina, pero ahora en el punto "B". La situación de la tabla después del paso 7 es la siguiente:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—
2	2	3	-1	3	—	—
3	3	0	4	4	—	añil
4	4	0	5	5	—	dorado
5	5	0	-2	2	—	gualdo
6	2	3	-1	1	sí	rojo
7	1	2	6	6	sí	verde

Nos encontramos en el nudo número 1, pero hemos entrado desde el punto B. Escribimos el contenido del registro (verde) y luego, como el puntero derecho no es negativo, transferimos su valor a "X" y volvemos al punto "A".

La situación final de la tabla de los estados es la siguiente:

Paso	X Primero	Izq(x)	Der(x)	X Después	Entrada en B	Impresión
1	1	2	6	2	—	—
2	2	3	-1	3	—	—
3	3	0	4	4	—	añil
4	4	0	5	5	—	dorado
5	5	0	-2	2	—	gualdo
6	2	3	-1	1	sí	rojo
7	1	2	6	6	sí	verde
8	6	10	0	0	—	violeta

Nos encontramos en el sexto nudo: el puntero izquierdo contiene el valor 0 y por ello imprimimos el contenido del registro (violeta). Puesto que el puntero derecho contiene el valor 0, habrá terminado el recorrido del árbol.

Una pequeña observación final es que la técnica de los punteros negativos "hacia atrás" no es general. No obstante, la hemos encontrado cómoda (además de que es eficiente porque abrevia las nuevas subidas de los nudos, que, de no ser así, serían necesarias) sobre todo en la realización de programas en BASIC.

CAPITULO III

LAS BASES DE DATOS

Fundamentos de las bases de datos



Se ha indicado con anterioridad que es frecuente que dos o más tipos de archivos, o ficheros, se procesen tanto de forma independiente como de forma conjunta entre sí.

Por ejemplo, el fichero de clientes, contiene informaciones tales como su código, el apellido, el nombre, el domicilio, el código fiscal, etc. Al mismo tiempo, otro fichero contiene los movimientos realizados por los clientes, así como las informaciones pertinentes: código del cliente, fecha, código de la mercancía, cantidad vendida, etc. Es fácil intuir que la actualización del fichero de clientes se realiza en tiempo y en modo diferentes con respecto a los del fichero de movimientos. El fichero de clientes se puede actualizar cuando se incluye un nuevo cliente o cuando tenemos que modificar los datos de alguno. El fichero de movimientos se actualiza de forma cotidiana, pero solamente para aquellos clientes que hicieron adquisiciones. Sin embargo, cuando imprimimos las facturas tenemos que trabajar con ambos ficheros.

Es también intuitivo pensar que no conviene duplicar los datos del cliente en los movimientos; sería un enorme desperdicio de espacio en disco y puede dar lugar a confusión si nos equivocamos y escribimos de manera diferente los datos en los dos ficheros.

La ventaja de la organización base de datos ("data base") consiste precisamente en el hecho de poner en relación varios ficheros evitando la duplicación de las informaciones.

Habida cuenta del objeto eminentemente divulgativo de esta

monografía, nos limitaremos a una base de datos de tipo didáctico, constituida por dos ficheros individuales y una sola clave de acceso. Para no molestar y no hacer que se produzcan sospechas de "connivencia" con la Renfe, la explicación correspondiente no se hará con la analogía de locomotoras y vagones, sino con otra más apta para aclarar estos conceptos.

Un ejemplo muy sencillo

Concibamos el fichero como si fuera un archivador con cajones, vacío si no contienen nada los registros. Cada registro está representado por un cajón. El primer cajón lleva el número 0; el segundo, el número 1, y así sucesivamente.

Cada cajón, o casillero, del fichero de "clientes" contiene dos compartimientos. El compartimiento de la izquierda contiene los datos del cliente. El compartimiento de la derecha contiene un puntero. Si el puntero es de valor -1 (o, en general, NIL), ello significa que no hay movimientos de ese cliente; en caso contrario indica en qué registro del fichero de "movimientos" se encuentran las adquisiciones o, dicho de otro modo, el número del registro que se debe procesar.

Cada casillero del fichero de "movimientos" contiene, a su vez, tres compartimientos: el de la izquierda contiene los datos correspondientes al movimiento, el central contiene el número del registro del fichero de "clientes" que debe leerse para saber el apellido del cliente, y, finalmente, el compartimiento de la derecha contiene un puntero. Si el puntero contiene el valor -1 quiere decir que no hay otros movimientos; en caso contrario, indica el próximo movimiento (en la práctica, el número del registro que debe procesarse para conocer la adquisición siguiente realizada por el mismo cliente).

El fichero de clientes se crea antes de efectuar cualquier movimiento. La figura 1 muestra la situación, estando vacíos los archivadores del fichero de clientes y del fichero de movimientos.

Se presenta ahora el señor "Verde" y realiza dos adquisiciones. No tenemos ningún cliente y, por consiguiente, tenemos que escribir sus datos personales; los escribimos en el fichero de clientes. Es fácil imaginar la situación después de la inscripción del cliente Verde: el casillero 0 contiene la clave "Verde" y, además, contiene el valor -1, que sirve para indicar que, de momento, no hay movimientos.

La figura 2 muestra, por el contrario, la situación después del registro de los dos movimientos realizados por el cliente Verde. Se han ocupado los dos casilleros 0 y 1 del fichero MOVIMIEN-

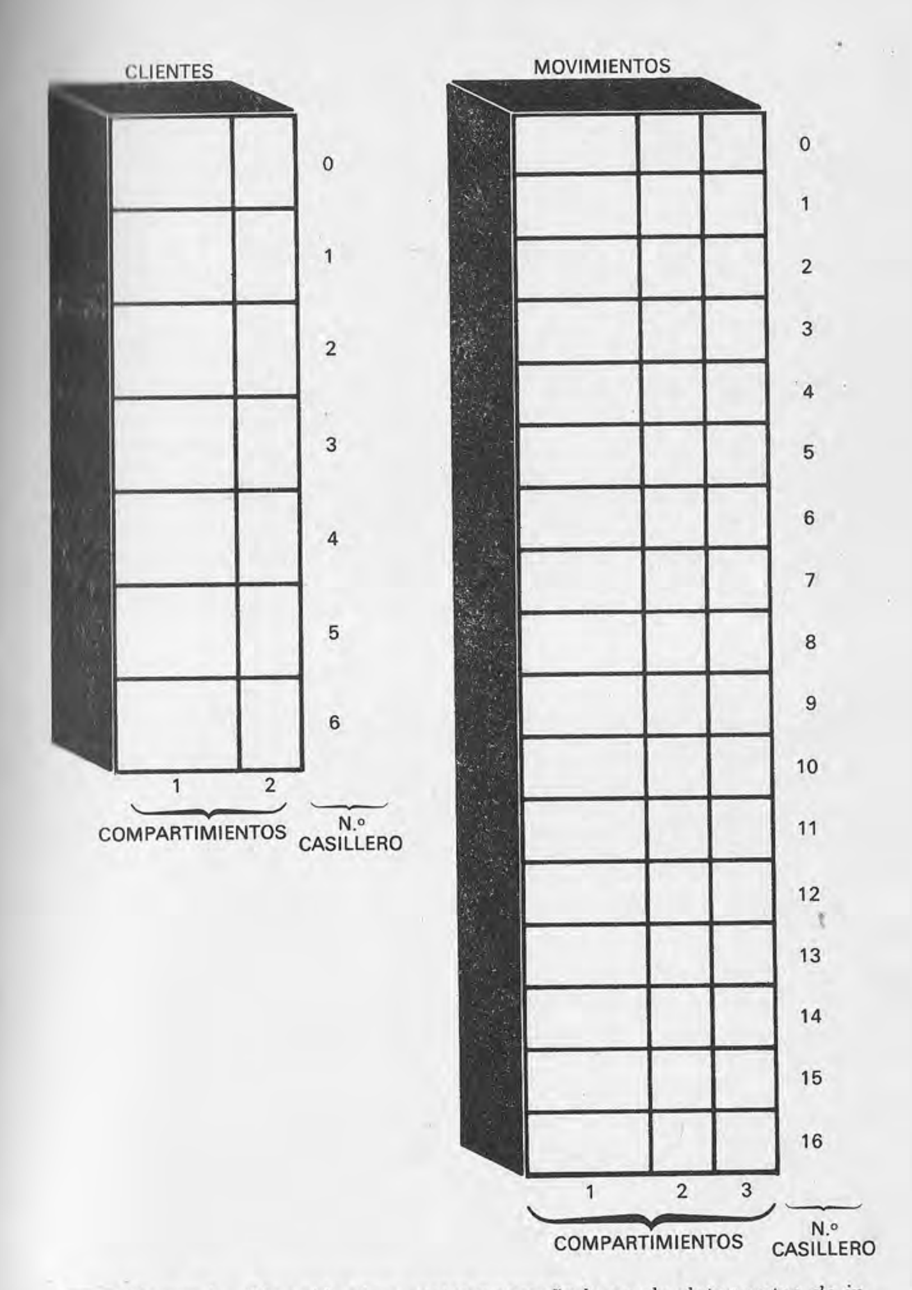


Figura 1.—Situación de nuestra pequeña base de datos antes de introducir cualquier registro. Los "archivadores" de CLIENTE y MOVIMIENTOS están vacíos.

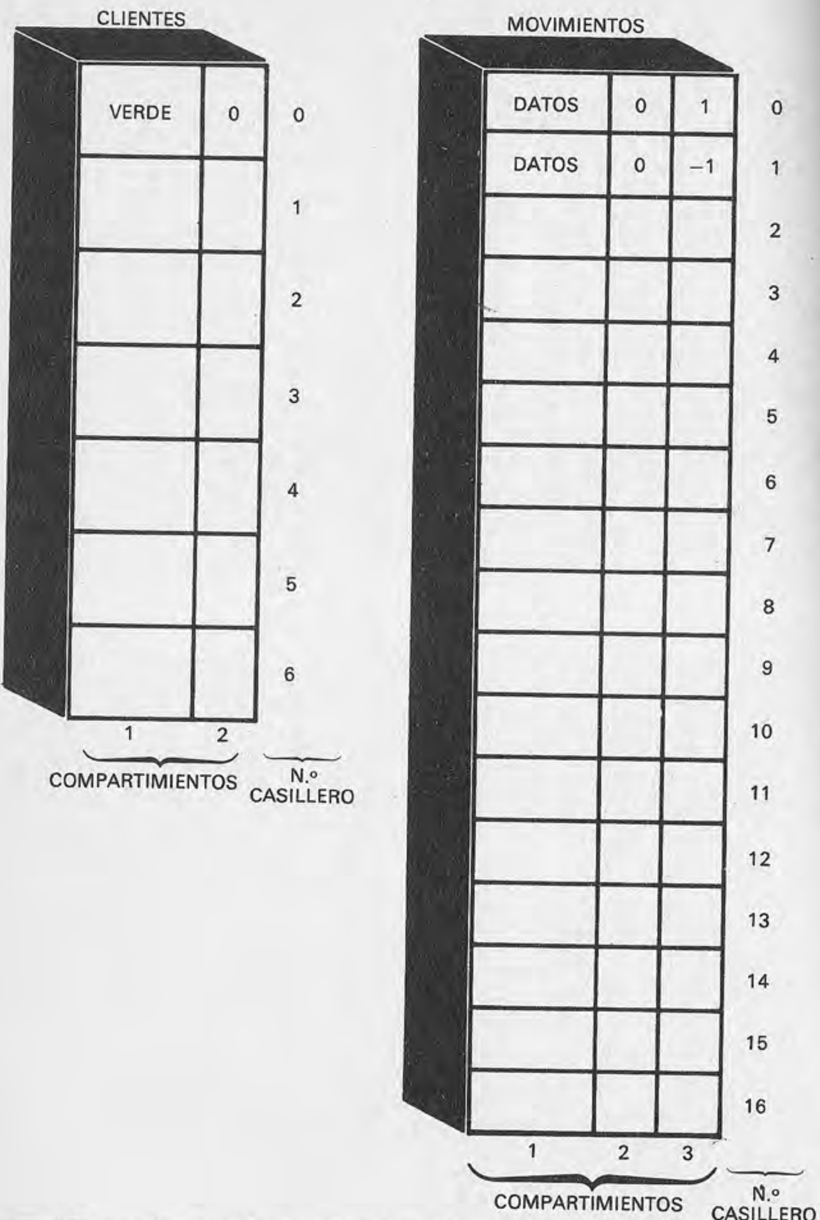


Figura 2.—Gestión de una base de datos: situación después de actualizar el registro de Verde y los movimientos asociados a él. El puntero de Verde tiene ahora el valor 0, mientras que antes de registrar los movimientos tenía el valor -1 (NIL - nada).

TOS. Cada casillero lleva la indicación genérica de "DATOS" (para los fines de nuestra simulación no es preciso saber nada más). Sabemos que cada casillero tiene tres compartimientos: en el primero se almacenan los "datos", el segundo indica en qué casillero del fichero "clientes" se encuentra la identificación del cliente y el tercer compartimiento indica si hay otros movimientos realizados por el cliente. En el último compartimiento del casillero 0 se encuentra el valor 1 y ello significa que el cliente ha hecho otra adquisición y que esta adquisición está descrita en el casillero 1. En el último compartimiento del casillero 1 encontramos el valor -1, lo que quiere decir que el cliente no ha efectuado otras adquisiciones.

En el archivador de clientes han cambiado también algunas cosas: el último compartimiento del casillero 0 ya no contiene "-1", sino el valor 0, lo que significa que el cliente Verde ha realizado adquisiciones y que su primer movimiento se encuentra en el casillero 0 del archivador de los movimientos.

Hemos dicho ya que antes de escribir los movimientos tendremos que escribir las características del cliente en el fichero de clientes. Por ello rogamos a la sección de ventas que nos proporcione inmediatamente los datos de los nuevos clientes. En condiciones normales, el cliente se inscribe de forma inmediata en el fichero, mientras que los movimientos se almacenan al final de la jornada de trabajo si se trabaja en el modo "batch", es decir, por lotes o por paquetes. Si, por el contrario, se trabaja en tiempo real, los clientes y los movimientos se registran de inmediato. En nuestra simulación haremos referencia al funcionamiento de tipo "batch". El departamento de ventas, en respuesta a nuestras exigencias, nos comunica los datos de tres nuevos clientes: Rojo, Añil y Dorado.

La figura 3 muestra la situación después de registrar a los tres nuevos clientes. Cada uno de estos tres clientes no tiene ningún movimiento por ahora. El departamento de ventas nos comunica luego que el cliente Rojo ha realizado una adquisición y que el cliente Dorado ha efectuado dos adquisiciones. El señor Añil no ha realizado ninguna adquisición, excusándose por haber olvidado el dinero en casa y prometiendo venir al día siguiente.

No es difícil imaginar la situación después del registro de la única adquisición realizada por el cliente señor Rojo. En el archivador de clientes, en el último compartimiento del casillero Rojo estará el valor 2, porque el primer movimiento realizado por el cliente se encuentra en el fichero de movimientos y, precisamente, en el casillero 2. En este casillero del fichero de movimientos encontramos: en el primer compartimiento, los datos correspondientes a lo que ha adquirido el cliente en el segundo compartimiento, la referencia a sus datos (es decir, lo relativo al casille-

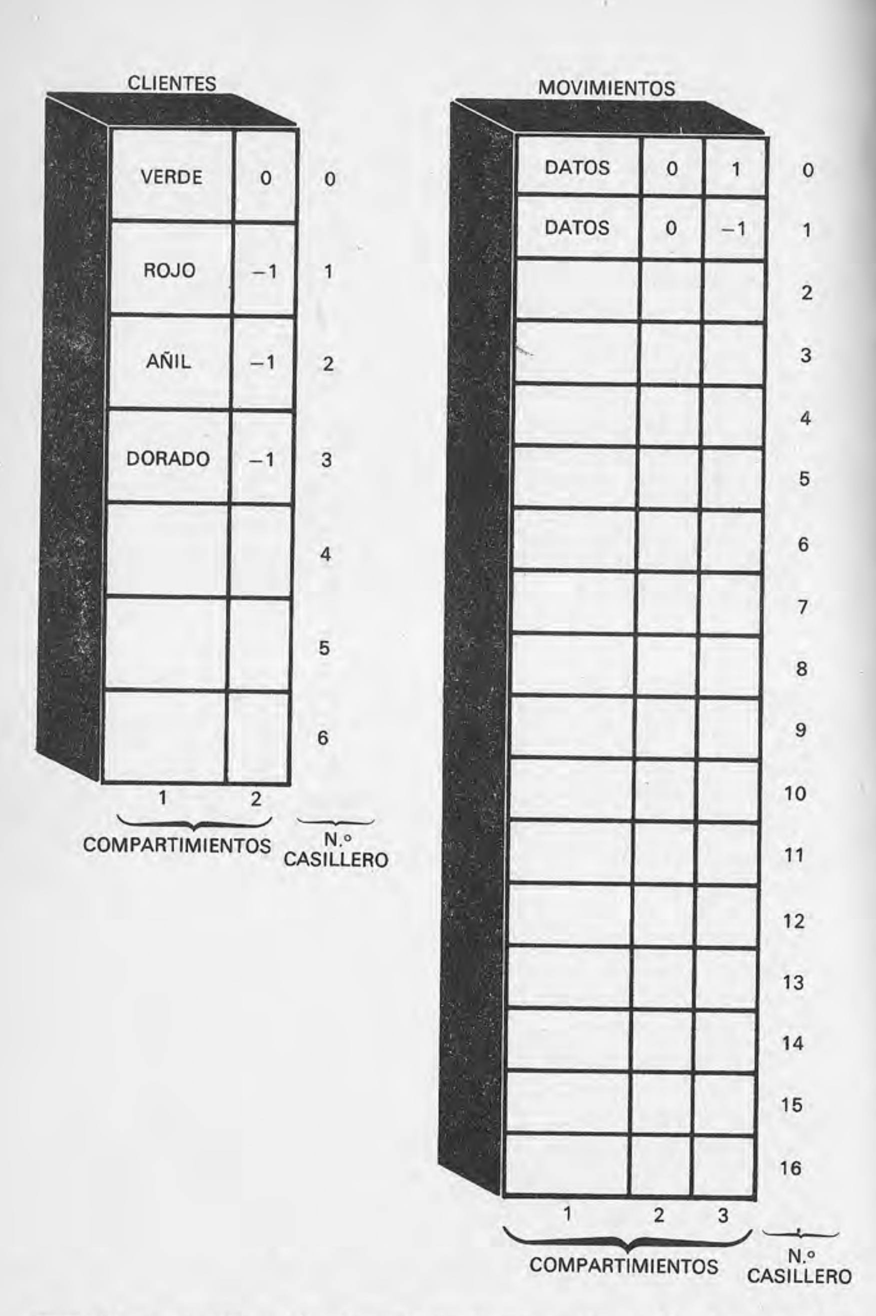


Figura 3.—Situación después de la grabación de los datos de tres nuevos clientes: Rojo, Añil y Dorado.

ro 1 del fichero de clientes), y en el tercer compartimiento hay la indicación de que no existen otros movimientos.

La figura 4 muestra la situación después de registrar también las dos adquisiciones realizadas por el cliente señor Dorado. En el fichero de clientes, el señor Añil sigue con el puntero en −1 (NIL - nada), que indica la falta de adquisiciones, mientras que el cliente Dorado tiene un puntero dirigido al casillero 3 del fichero de movimientos. En el fichero de movimientos, el casillero número 3 contiene: en el segundo compartimiento, la indicación de que el nombre del cliente (Dorado) se encuentra en el casillero número 3 del fichero de clientes, en el primer compartimiento se indica lo que ha adquirido y en el tercer compartimiento se indica que la siguiente adquisición se encuentra registrada en el casillero número 4. En el último compartimiento del casillero número 4 está el valor −1, porque no hay otras adquisiciones.

El departamento de ventas nos avisa luego de haber captado a los clientes Gualdo y Violeta. Dejamos como fácil ejercicio al lector la representación gráfica de dicha situación, así como de la posterior a la comunicación habitual del departamento de ventas, según la cual el señor Gualdo ha efectuado una sola adquisición.

El movimiento se registrará en el casillero número 5, cuyo último compartimiento nos indicará que no existen otros movimientos. En el fichero de clientes, el último compartimiento del casillero 4 nos indica que el primer movimiento se encuentra en el casillero 5 del fichero de movimientos.

Al final de la jornada el departamento de ventas nos avisa de que el cliente Violeta ha realizado una adquisición. La figura 5 muestra la situación después de esta última actualización de los dos ficheros y teniendo en cuenta también lo que se acaba de indicar. En el fichero de clientes, el puntero del cliente Violeta indica que la primera adquisición se encuentra en el sexto casillero del fichero de movimientos. En el fichero de movimientos, el último compartimiento del casillero 6 nos informa que el cliente Violeta no tiene ningún otro movimiento.

Más actualizaciones

Dijimos ya que la actualización no se suele realizar de forma inmediata, sino de forma periódica, por ejemplo al final de la jornada de trabajo o al final de la semana. Al final de la jornada, el departamento de ventas nos avisa que el señor Verde ha hecho otra adquisición, que el señor Rojo hizo tres, el señor Añil realizó una, el señor Dorado otra, el señor Gualdo efectuó dos adquisiciones y el señor Violeta un movimiento. Sabemos que los movimien-

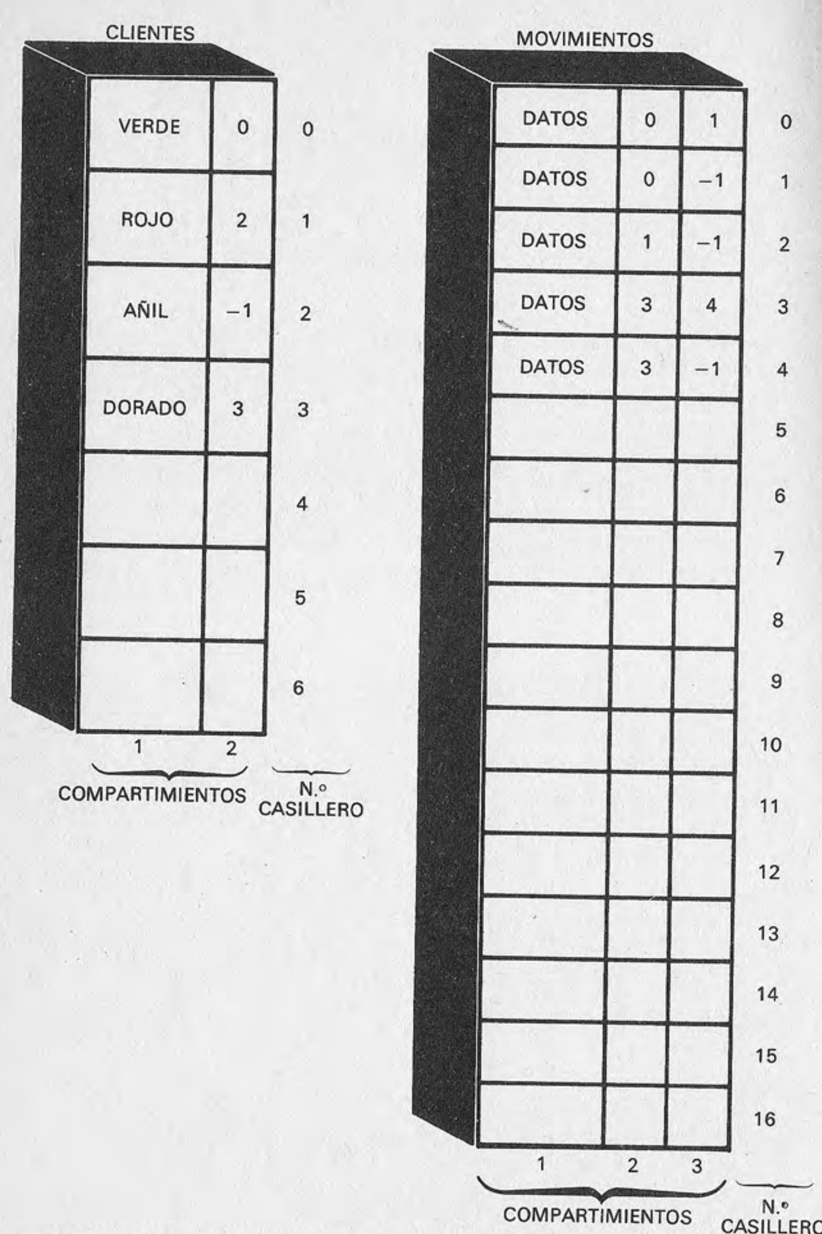


Figura 4.—Situación después de la grabación de los movimientos de los clientes Rojo y Dorado. El cliente Añil no ha hecho todavía ninguna adquisición y esta circunstancia le indica el puntero -1 (NIL - nada) del fichero CLIENTES.

CLIENTES

Compartimiento 1	Compartimiento 2	N.º casillero
VERDE	0	0
ROJO	2	1
AÑIL	−1	2
DORADO	3	3
GUALDO	5	4
VIOLETA	6	5
		6

MOVIMIENTOS

Compartimiento 1	Compartimiento 2	Compartimiento 3	N.º casillero
DATOS	0	1	0
DATOS	0	−1	1
DATOS	1	−1	2
DATOS	3	4	3
DATOS	3	−1	4
DATOS	4	−1	5
DATOS	5	−1	6
			7
			8
			9
			10
			11
			12
			13
			14
			15
			16

Figura 5.—Situación de la base de datos inmediatamente después de la grabación del movimiento del cliente Violeta.

tos se colocan "a la cola" de los registros ya existentes en el fichero de movimientos. En la práctica, una vez que se haya llenado hasta el casillero 6, llenaremos los casilleros desde el número 7 en adelante. ¿Sucederá lo mismo en el fichero de los clientes? La respuesta es afirmativa, tal como se verá de inmediato.

La figura 6 nos muestra la situación después de registrar el nuevo movimiento del cliente Verde. Lo primero que hay que hacer es abrir el casillero 0 del fichero de clientes y tomar nota del contenido de su último compartimiento (es decir cero) (Fig. 5). A continuación, se depositan los datos en el casillero 7 del fichero de movimientos y en el segundo compartimiento del mismo casillero se pone cero, que es la clave que sirve para indicar el lugar del nombre en el registro de clientes. En el tercer compartimiento ponemos de forma análoga cero, que es el valor que hemos encontrado en el último compartimiento del fichero de clientes. Finalmente, en este último compartimiento mencionado del fichero de clientes ponemos el valor 7, que es el número del casillero del fichero de movimientos en donde se ha colocado el movimiento correspondiente.

De este modo, si abrimos el casillero 0 del fichero de clientes sabemos que el cliente se llama Verde y que una de sus adquisiciones se encuentra en el casillero 7 del fichero de movimientos. Abrimos este casillero y con el contenido de su último compartimiento nos dirigimos al casillero 0 del mismo fichero. Abrimos el 0 y el último compartimiento nos indica que hay otro movimiento, precisamente en el casillero 1. Finalmente, abrimos el casillero 1 y encontramos la adquisición; en el último compartimiento se nos indica que no hay otros movimientos.

La figura 7 nos muestra la situación después de registrar los tres movimientos del cliente Rojo. Los movimientos se han introducido en los casilleros 8, 9 y 10; en el último compartimiento de cada casillero está el puntero al siguiente movimiento. Debemos prestar atención al último compartimiento del casillero 10, que contiene el valor 2, tomado del último compartimiento del casillero del cliente Rojo antes de que fuera actualizado. El último compartimiento del casillero Rojo en el fichero de clientes tiene como puntero el valor 8. A costa de resultar pesados, veamos con detalle otras situaciones con la ilustración correspondiente.

La figura 8 muestra la situación después de registrar la adquisición hecha por el cliente Añil. También en este caso el contenido anterior del último compartimiento del casillero 2 (de Añil) se ha colocado en el último compartimiento de la adquisición almacenada en el casillero 11 del fichero de movimientos. Intenten ustedes acabar las demás actualizaciones.

La figura 9 muestra la situación después de registrar la adquisición realizada por el cliente Dorado.

CLIENTES

COMPARTIMIENTO 1	COMPARTIMIENTO 2	N.º CASILLERO
VERDE	7	0
ROJO	2	1
AÑIL	−1	2
DORADO	3	3
GUALDO	5	4
VIOLETA	6	5
		6

MOVIMIENTOS

COMPARTIMIENTO 1	COMPARTIMIENTO 2	COMPARTIMIENTO 3	N.º CASILLERO
DATOS	0	1	0
DATOS	0	−1	1
DATOS	1	−1	2
DATOS	3	4	3
DATOS	3	−1	4
DATOS	4	−1	5
DATOS	5	−1	6
DATOS	0	0	7
			8
			9
			10
			11
			12
			13
			14
			15
			16

Figura 6.—Situación después de la grabación de otro movimiento del cliente Verde.

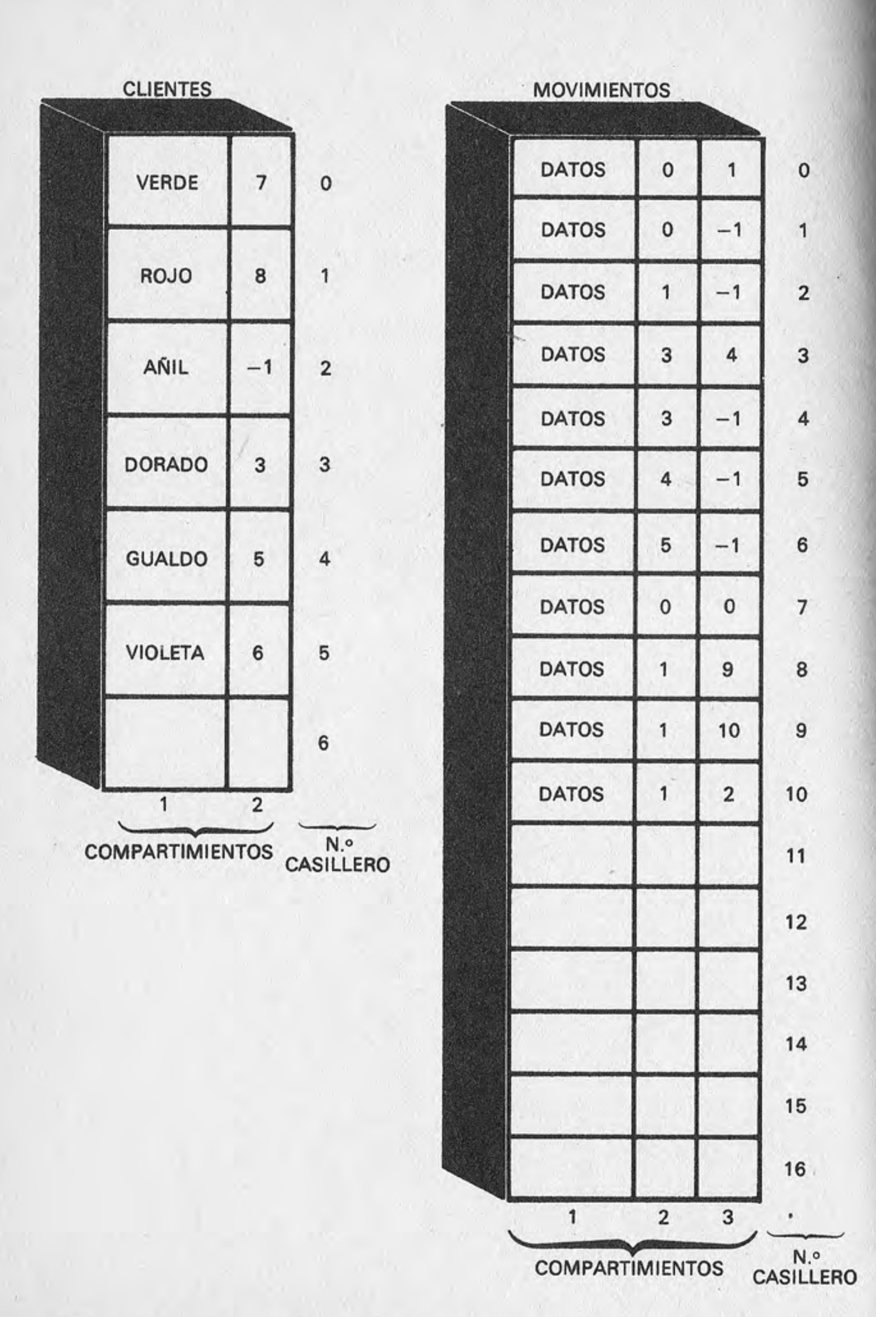


Figura 7.—Situación después de la grabación de otros tres movimientos del cliente Rojo.

CLIENTES

1	2	N.º CASILLERO
VERDE	7	0
ROJO	8	1
AÑIL	11	2
DORADO	3	3
GUALDO	5	4
VIOLETA	6	5
		6

COMPARTIMIENTOS

MOVIMIENTOS

1	2	3	N.º CASILLERO
DATOS	0	1	0
DATOS	0	−1	1
DATOS	1	−1	2
DATOS	3	4	3
DATOS	3	−1	4
DATOS	4	−1	5
DATOS	5	−1	6
DATOS	0	0	7
DATOS	1	9	8
DATOS	1	10	9
DATOS	1	2	10
DATOS	2	−1	11
			12
			13
			14
			15
			16

COMPARTIMIENTOS

Figura 8.—Situación después de la grabación de un movimiento del cliente Añil.

CLIENTES

Compartimiento 1	Compartimiento 2	N.° casillero
VERDE	7	0
ROJO	8	1
AÑIL	11	2
DORADO	12	3
GUALDO	5	4
VIOLETA	6	5
		6

MOVIMIENTOS

Compartimiento 1	Compartimiento 2	Compartimiento 3	N.° casillero
DATOS	0	1	0
DATOS	0	−1	1
DATOS	1	−1	2
DATOS	3	4	3
DATOS	3	−1	4
DATOS	4	−1	5
DATOS	5	−1	6
DATOS	0	0	7
DATOS	1	9	8
DATOS	1	10	9
DATOS	1	2	10
DATOS	2	−1	11
DATOS	3	3	12
			13
			14
			15
			16

Figura 9.—Situación después de la grabación de otro movimiento del cliente Dorado.

La figura 10 muestra la situación después de actualizar las posteriores adquisiciones realizadas por el cliente Gualdo.

La figura 11 muestra la situación después del registro del último movimiento realizado por el cliente Violeta.

De forma intencionada, no hemos hablado hasta ahora de cómo se realiza la búsqueda del nombre del cliente en el fichero correspondiente, con el objeto de no hacer más farragosa todavía la explicación.

Qué hacer con las bases de datos

En condiciones normales, para tener acceso a las claves se utiliza la técnica explicada con anterioridad en el acceso de índices, es decir, las claves se almacenan en una tabla ordenada en sentido ascendente. A menudo se emplea el árbol binario para almacenarlas.

¿Cómo podemos utilizar y sacar provecho de esta base de datos que estamos creando? Se pueden efectuar en ella diversas operaciones:

- consultar solamente el fichero de clientes;
- consultar solamente el fichero de movimientos;
- consultar simultáneamente los dos ficheros;
- actualizar, en el sentido más amplio, los dos ficheros.

Consultando el fichero de clientes podemos tener todas las informaciones sobre ellos y, además, saber de cada uno de los clientes si efectuó o no al menos una adquisición. El fichero de clientes podemos consultarlo tanto por el orden de inscripción del cliente como por el orden alfabético de sus apellidos. Podemos modificar los datos existentes en el registro y podemos añadir nuevos clientes o suprimir algunos.

Hagamos de nuevo referencia a la figura 11. Si leemos el fichero de clientes de forma secuencial, tendremos a nuestra disposición los clientes en el orden siguiente: Verde, Rojo, Añil, Dorado, Gualdo, Violeta. Si leemos el fichero de clientes con la ayuda de la clave dispondremos de los clientes en orden alfabético: Añil, Dorado, Gualdo, Rojo, Verde y Violeta. Tanto en uno como en otro caso, al observar el último compartimiento de cada casillero podemos saber si el cliente ha realizado adquisiciones o no y, en caso afirmativo, dónde encontrar los datos de éstas.

Al consultar el fichero de movimientos podemos saber cuántas adquisiciones se han realizado y en el orden en que se efectuaron. Podemos saber si la adquisición fue hecha por un cliente o por otro, y para averiguar su nombre basta consultar el fichero

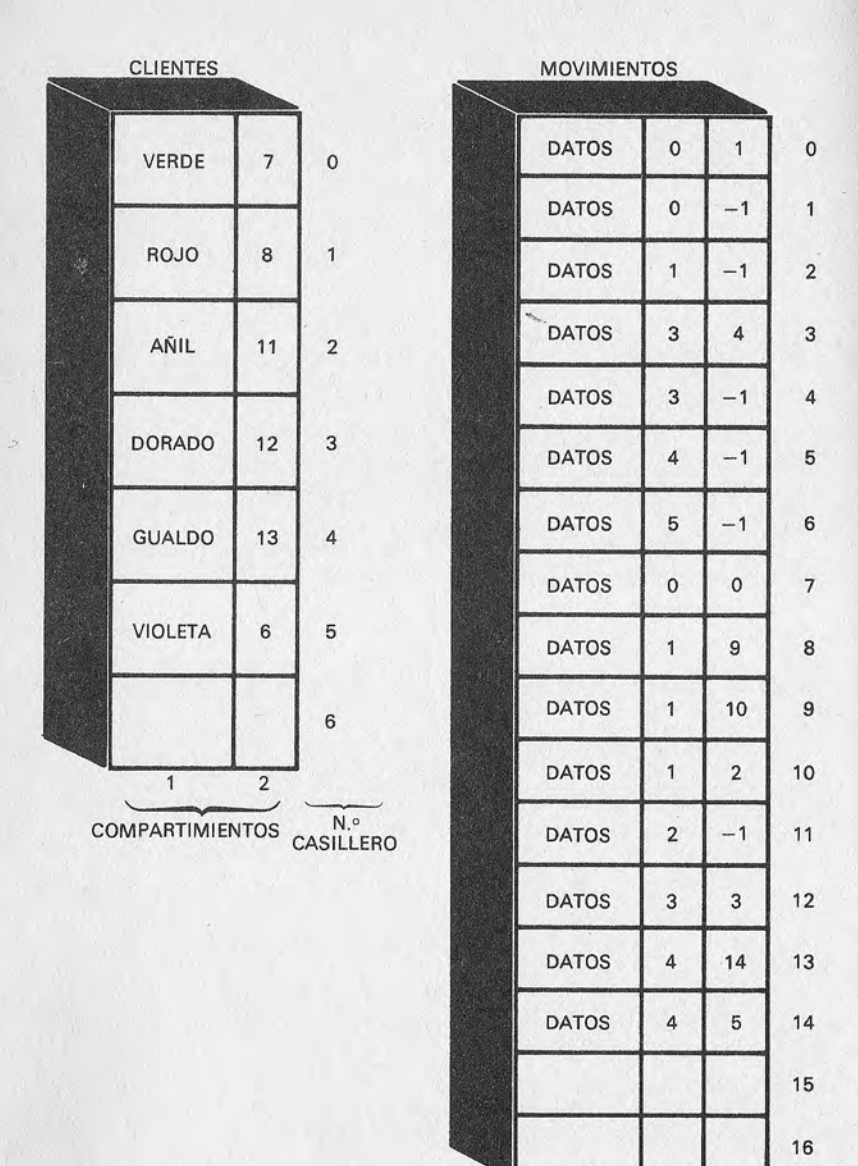


Figura 10.—Situación después de la grabación de otros dos movimientos del cliente Gualdo.

CLIENTES

COMPARTIMIENTO 1	COMPARTIMIENTO 2	N.º CASILLERO
VERDE	7	0
ROJO	8	1
AÑIL	11	2
DORADO	12	3
GUALDO	13	4
VIOLETA	15	5
		6

MOVIMIENTOS

COMPARTIMIENTO 1	COMPARTIMIENTO 2	COMPARTIMIENTO 3	N.º CASILLERO
DATOS	0	1	0
DATOS	0	−1	1
DATOS	1	−1	2
DATOS	3	4	3
DATOS	3	−1	4
DATOS	4	−1	5
DATOS	5	−1	6
DATOS	0	0	7
DATOS	1	9	8
DATOS	1	10	9
DATOS	1	2	10
DATOS	2	−1	11
DATOS	3	3	12
DATOS	4	14	13
DATOS	4	5	14
DATOS	5	6	15
			16

Figura 11.—Situación después de la grabación de otro movimiento del cliente Violeta.

de clientes. Podemos añadir otras adquisiciones, modificar los datos de un registro y suprimir adquisiciones. Haciendo siempre referencia a la figura 11 y leyendo de forma secuencial el fichero de movimientos obtenemos la lista de las adquisiciones en el orden en que se efectuaron y, sin necesidad de consultar el fichero de clientes, podemos saber si se efectuaron o no por el mismo cliente.

Si queremos consultar los dos ficheros juntos debemos tomar como guía (maestro) uno de los dos. Si suponemos como guía el fichero de los clientes podemos saber qué adquisiciones hizo cada cliente. Por el contrario, si tomamos como guía el fichero de movimientos, podemos saber para cada movimiento el nombre del cliente que lo efectuó. La figura 11 nos indica también cómo se puede operar, por ejemplo, para saber la situación completa del cliente Gualdo. En la tabla de las claves encontramos la referencia 4 para él. Abrimos el casillero 4 del fichero de clientes y encontramos allí los datos del cliente y, además, un puntero que contiene el valor 13 y que está dirigido al fichero de movimientos. A continuación nos dedicamos a éste. Abrimos el casillero número 13 y encontramos los datos correspondientes a una adquisición, y luego, en el último compartimiento, encontramos el valor 14. Abrimos el casillero 14, encontramos los datos correspondientes a otra adquisición, y en el último compartimiento encontramos el valor 5. Abrimos el casillero 5 y encontramos los datos correspondientes a otra adquisición. Finalmente, en el último compartimiento, encontramos el valor -1, que indica que no se efectuaron más adquisiciones. De este modo se buscan los movimientos realizados en la última actualización, luego los de la penúltima actualización (en nuestro caso, primero los datos de la segunda actualización y luego los de la primera) y así sucesivamente en sentido ascendente.

La figura 12 muestra la situación después de anular la tercera adquisición realizada por el cliente Rojo. Para ello abrimos el casillero 1 del fichero de clientes y en el último compartimiento encontramos que sus adquisiciones han de buscarse en el fichero de movimientos a partir del casillero 8. Abrimos el casillero 8 del fichero de movimientos y su último compartimiento nos indica que el siguiente casillero es el número 9. Abrimos el casillero número 9 y en su último compartimiento está la indicación de que la siguiente adquisición se encuentra en el casillero 10 y, además, éste es el registro que queremos eliminar. Bastará a este respecto, para indicar que el registro se ha anulado, que introduzcamos el valor -1 en el compartimiento del cliente. La anulación está concluida; como se dijo con anterioridad, no hemos previsto la recuperación del espacio libre. En general, pues, para anular un movimiento bastará poner un -1 en el código del cliente.



MOVIMIENTOS

Compartimiento 1	2	3	N.º casillero
DATOS	0	1	0
DATOS	0	−1	1
DATOS	1	−1	2
DATOS	3	4	3
DATOS	3	−1	4
DATOS	4	−1	5
DATOS	5	−1	6
DATOS	0	0	7
DATOS	1	9	8
DATOS	−1	10	9
DATOS	1	2	10
DATOS	−1	−1	11
DATOS	3	3	12
DATOS	4	14	13
DATOS	4	5	14
DATOS	5	6	15
			16

COMPARTIMIENTOS — N.º CASILLERO

Figura 12.—Situación después de la anulación de un movimiento del cliente Rojo.

CLIENTES

Compartimiento 1	Compartimiento 2	N.º casillero
VERDE	7	0
ROJO	8	1
AÑIL	11	2
DORADO	12	3
GUALDO	13	4
VIOLETA	15	5
		6

MOVIMIENTOS

Compartimiento 1	Compartimiento 2	Compartimiento 3	N.º casillero
DATOS	0	1	0
DATOS	0	−1	1
DATOS	1	−1	2
DATOS	3	4	3
DATOS	3	−1	4
DATOS	4	−1	5
DATOS	5	−1	6
DATOS	0	0	7
DATOS	1	9	8
DATOS	−1	10	9
DATOS	1	2	10
DATOS	2	−1	11
DATOS	3	3	12
DATOS	4	14	13
DATOS	4	5	14
DATOS	5	6	15
			16

Figura 13.—Situación después de la anulación de un movimiento del cliente Añil.

Finalmente, la figura 13 muestra la situación después de la anulación de la única adquisición realizada por el cliente Añil. Abrimos el casillero del cliente Añil y en el último compartimiento encontramos el valor 11. Abrimos el casillero 11 del fichero de movimientos y en el último compartimiento encontramos el valor -1, que corresponde al hecho de que no hay otras adquisiciones. Para indicar que el registro fue anulado, en el segundo compartimiento, en el lugar de la referencia al cliente, ponemos el valor -1 y cerramos el casillero. En el último compartimiento del casillero del cliente Añil ponemos el valor -1, extraído del último compartimiento del casillero de movimientos y, de este modo, sabemos que no hay otros movimientos (entre nosotros, un cliente de este género vale la pena suprimirlo del archivo, pues no es nada rentable).

Debería ser evidente, pero, no obstante, hemos de remarcar el hecho de que antes de eliminar un cliente del archivo deberán eliminarse TODOS sus movimientos. La eliminación se suele efectuar suprimiendo la entrada correspondiente en la tabla de las claves y señalando el registro que se ha anulado.

El sistema que hemos utilizado para actualizar los ficheros tiene el inconveniente de que cuando queramos conocer los movimientos efectuados por cada cliente éstos no se presentarán en el orden en que se efectuaron. Por supuesto, es posible actuar de modo que se tengan los movimientos en el orden exacto en que se realizaron. Sin embargo, no hemos querido elegir este camino porque la técnica adoptada es la más rápida y sencilla, por cuanto que limita las operaciones de entrada/salida.

CAPITULO IV

PROGRAMAS Y SUBRUTINAS PARA GESTIONAR FICHEROS

Especificaciones

Comenzamos ahora la presentación y comentario de los primeros programas para la gestión de ficheros como los que hemos visto en los ejemplos de capítulos anteriores; en nuestra segunda monografía sobre bases de datos abordaremos los casos más complejos. Trataremos, sobre todo, de eliminar las dudas e inconvenientes puestos de manifiesto; presentaremos, además, programas para la recuperación del espacio debido a anulación de los registros. En los ficheros de acceso directo utilizaremos el primer registro para mantener algunas informaciones, mientras que emplearemos los registros sucesivos para conservar los datos del fichero. En el PASCAL UCSD los registros de un fichero de acceso aleatorio están numerados del 0 al 32767 y en el BASIC MS-DOS están numerados del 1 al 32767; en un intento de simplificar esta cuestión y no introducir elementos que despisten, en UCSD no utilizaremos el registro 0.

En los programas especiales, prepararemos el fichero con todos los campos adecuadamente puestos a cero y dispondremos el espacio en disco para su máxima amplitud. De este modo estableceremos un poco de orden en nuestros archivos y tomaremos precauciones ante la posibilidad de que otros programas nos "roben" el espacio en disco.

En el primer registro almacenamos dos informaciones: cuántos registros podrá contener el fichero y cuál será el primer registro a grabar.

Además, hemos de tener presente que una vez asignado el

espacio en disco no tendrá ningún significado el empleo de EOF (final de fichero). Si queremos utilizar esta técnica, tendremos que escribir un programa que configure de manera adecuada el archivo.

Para la gestión de las claves no utilizaremos tablas en memoria, sino otro fichero. Por consiguiente, diremos adiós a los "armarios" y a los "vagones". En algunos casos utilizaremos para las claves el árbol binario.

Presentaremos los programas tanto en BASIC MS-DOS como en PASCAL UCSD y los nombres de las subrutinas serán idénticos. Todos los programas se han incluido en el apéndice, en donde, en un intento de ser más claros, las llamadas a las subrutinas (GOSUB) se han destacado de forma adecuada. Antes de adentrarnos en las diversas organizaciones, presentamos algunas subrutinas válidas en todos los tipos de fichero.

Subrutina de borrado de pantalla

Esta subrutina es muy sencilla y sólo hemos pretendido presentarla con el fin de que sea la primera en adaptarse al propio hardware. Lamentablemente, para borrar los caracteres en la pantalla se utilizan comandos que varían de un lenguaje a otro y de un ordenador a otro. Consultando los manuales correspondientes no resulta difícil identificar los caracteres de control que deben utilizarse. Lo importante, a nuestro juicio, es destacar la instrucción que debe modificarse y que ésta sea una sola (éste es precisamente el objeto de la subrutina).

Para el lenguaje BASIC MS-DOS la subrutina está incluida en el apéndice y se repite para cada caso. La subrutina ocupa siempre las líneas 500 a 550.

```
540 CLS
550 RETURN
```

También para el PASCAL UCSD las instrucciones correspondientes al procedimiento BORRADO se llevan al apéndice. A título excepcional damos en esta ocasión el procedimiento completo.

```
PROCEDURE borrado de pantalla;
BEGIN
    WRITE    (CHR(28));
END;
```


Subrutina del teclado

Cuando se introducen datos a través del teclado es conveniente comprobar la validez de dichos datos. Por ejemplo, los campos numéricos deben estar constituidos solamente por dígitos. Si ante la solicitud de datos numéricos se responde con una tecla alfabética, tanto en BASIC como en Pascal se emitirá un mensaje de error. Ello resulta antipático e incómodo, sobre todo si se tiene la buena costumbre de guiar al operador en la introducción de los datos presentando en la pantalla una "máscara". Los mensajes de error perturban la máscara y, en Pascal, estos mensajes interrumpen el programa.

La subrutina prevé la introducción de datos del tipo INTEGER (entero), LONG INTEGER (para el Pascal), COMA FLOTANTE (para el BASIC) o CADENAS. La variable SW indica el tipo de dato. Las variables XPOS e YPOS indican en qué línea y columna debe darse la respuesta. La subrutina prevé el control de la tecla (flecha a la izquierda, retroceso —backspace— o CONTROL-Q, según el ordenador que utilicemos), adecuada para corregir los caracteres ya escritos.

En los ejemplos que presentamos, el valor binario de la tecla de borrado (backspace) es 8 para el Pascal y 29 para el BASIC. La subrutina continúa leyendo caracteres hasta que se pulse la tecla <RETURN>. Si el campo es numérico y no se están empleando las teclas del 0 al 9, se emitirá una señal acústica. Si el campo es alfabético y no se utilizan las teclas adecuadas se producirá también una indicación acústica.

Si la variable SW se ha puesto a "1", entonces el campo se considerará numérico y el valor se restituirá a la variable PEQUEÑO. Si la variable SW se ha puesto a "2", el campo se considerará numérico, pero el valor correspondiente se restituirá en la variable GRANDE. Finalmente, si la variable SW se ha puesto a "3", el campo se considerará alfabético y el resultado se restituirá en la variable ALFA.

En BASIC, la subrutina se lleva al apéndice y, en las instrucciones 1000/1300, con el propósito de no complicar la vida a nuestros amables lectores, se repetirá en cada estructura. Comentemos las instrucciones siguientes:

```
1040 ALFA$=""
1050 GOSUB 15800:REM --->POSICIONA EL CURSOR
1060 GOSUB 15900:REM --->LECTURA DEL CARACTER
1070 IF ASC(CAR$)=13 THEN GOTO 1210
1080 IF ASC(CAR$)<>8 THEN GOTO 1310
```

La subrutina no está acabada y la hemos interrumpido momentáneamente para poderla comentar. Después de haber puesto a cero la variable ALFA$, cedemos el control a la subrutina de "lectura de carácter", que se encarga de leer un carácter a partir del teclado. En MS-DOS, para leer un carácter del teclado haremos:

```
15940 CAR$=INPUT$(1)
```

¿No le parece sencillo? Hemos querido aislar dicha instrucción porque varía de una versión de BASIC a otra. Una vez leído un carácter y cargado en la variable CAR$, examinaremos su contenido y

- si su valor corresponde al de la tecla de retorno (13) cederemos el control a la instrucción 1210;
- si el valor es 8 (backspace) cederemos el control a la instrucción 1120, que se encargará de borrar el carácter correspondiente;
- si el valor no corresponde ni a la tecla de retorno ni a la de corrección, el control pasará a la instrucción 1310.

Veamos la parte correspondiente al borrado:

```
1120 IF LEN(ALFA$)>1 THEN ALFA$=LEFT$(ALFA$,
     LEN (ALFA$)-1) ELSE ALFA$=""
1130 REM ---------------------------
1140 REM --  IMPRESION DEL CAMPO  --
1150 REM ---------------------------
1160 GOSUB 15800:REM -->POSICIONA EL CURSOR
1170 PRINT ALFA$;" ";
1180 GOSUB 15800:REM -->POSICIONA EL CURSOR
1190 PRINT ALFA$
1200 GOTO 1060
```

Esta segunda parte de la subrutina tiene como objeto borrar el último carácter introducido a través del teclado; también se aprovecha en parte en otros sitios de la subrutina. Hay que destacar la instrucción 1170, que escribe los caracteres introducidos a través del teclado seguidos por un espacio en blanco. La instrucción 1180 reposiciona el cursor y la instrucción 1190 vuelve a escribir los caracteres válidos introducidos a través del teclado de tal modo que el cursor quede posicionado de forma exacta.

Cuando se detecta un retorno de carro se activa la subrutina:

```
1210 REM ::::::::::::::::::::
1220 REM ::  NUEVA ENTRADA  ::
1230 REM ::::::::::::::::::::
1240 IF SW=3 THEN RETURN
1250 G#=0
1260 FOR X=1 TO LEN(ALFA$)
1270 G#=G#*10+ASC(MID$(ALFA$,X,1))-48
1280 NEXT X
1290 IF SW=1 THEN PEQUEÑO$=CSNG(G#) ELSE
     GRANDE#=G#
1300 RETURN
```

Con estas instrucciones se vuelve a la subrutina que efectuó la llamada. Si SW se hubiera puesto a 3 (estamos en un campo alfabético) se realiza el retorno de forma inmediata y, de no ser así, se tendrá que convertir el campo de cadena en un campo numérico. Las instrucciones 1260 a 1280 se encargan de rellenar el campo numérico G#. Algunas versiones del lenguaje BASIC tienen la posibilidad de convertir la cadena en un número; las instrucciones que presentamos pueden utilizarse en muchas versiones de BASIC.

Por último, si es un carácter normal, la subrutina será:

```
1310 REM -----------------------
1320 REM --  CARACTER NORMAL  --
1330 REM -----------------------
1340 IF SW=3 THEN GOTO 1450
1350 REM -------------------------
1360 REM --  CARACTER NUMERICO  --
1370 REM -------------------------
1380 IF CAR$<"0" OR CAR$>"9" THEN GOTO 1400
1390 GOTO 1490
1400 REM ----------------------
1410 REM --  ERROR DE TECLA  --
1420 REM ----------------------
1430 PRINT CHR$(7);
1440 GOTO 1060
1450 REM ---------------------------
1460 REM --  CARACTER ALFABETICO  --
1470 REM ---------------------------
```

```
1480 IF CAR$<" " OR CAR$>"z" THEN GOTO 1400
1490 ALFA$=ALFA$+CAR$
1500 GOTO 1130
```

La subrutina está terminada. Esta parte final se encarga de procesar los caracteres válidos; si el carácter es numérico debe tener un valor comprendido entre 0 y 9 (ver instrucción 1380) y, de no ser así, deberá tener un valor comprendido entre el espacio y "z" (instrucción 1480). Si el carácter no está dentro de los límites indicados, se emitirá una señal acústica, no se imprimirá el contenido de CAR$ y se pedirá otro carácter del teclado. Si el carácter es válido, se transferirá a la cola de la variable ALFA$, se escribirá y se solicitará el siguiente carácter del teclado.

La subrutina en PASCAL UCSD se encuentra en el apéndice y está inserta en cada programa de las estructuras que hemos visto. Como es habitual destacaremos las instrucciones más significativas:

```
CASE sw OF
      '1':pequeno:=0;
      '2':grande:=0;
      '3':alfa:=0;
END;
```

Ponemos a cero las variables PEQUEÑO, GRANDE y ALFA sobre la base del contenido de la variable SW.

COTOXY (POSX, POSY);

Situamos el cursor en las coordenadas indicadas por las variables POSX y POSY. Sigue luego un bucle de instrucciones que se repiten hasta que se pulse la tecla RETURN (UNTIL EOLN(KEYBOARD)). Se actúa sobre el dispositivo KEYBOARD (TECLADO), que tiene la particularidad de no escribir en la pantalla el carácter introducido a través del teclado o, como se dice técnicamente, que no hace eco.

```
REPEAT
  READ(KEYBOARD,car);
  IF NOT EOLN(KEYBOARD)
     THEN BEGIN IF ORD(car)=8 THEN BEGIN
          GOTOXY(x_pos,y_pos);
          CASE sw OF
             '1':BEGIN
```



```
                pequeno:=pequeno DIV 10;
                WRITE(pequeno,' ');
                GOTOXY(x_pos,y_pos);
                WRITE(PEQUENO);
             END;
```

Se lee un carácter del teclado y se compara, en este caso con el valor 8, es decir, la tecla de retorno (se tiene que volver a escribir el campo sin el último carácter). Según que el campo sea del tipo integer (entero), long integer (en Pascal) o string (cadena), se utilizan diferentes instrucciones. En el caso de variable entera (integer) para eliminar el último dígito se ha dividido PEQUEÑO por 10 y luego se escribe junto con un espacio en blanco para eliminar de la pantalla el carácter a borrar. El segundo WRITE sirve para posicionar adecuadamente el cursor. Continuemos la explicación de la subrutina.

```
'2':BEGIN
     grande:=grande DIV 10;
     WRITE(grande,' ');
     GOTOXY(x_pos,y_pos);
     WRITE(grande);
   END;
```

En el caso de "long integer", ejecutamos instrucciones muy similares a las empleadas para los campos "integer", actuando sobre la variable GRANDE en lugar de sobre la variable PEQUEÑO.

```
     '3':BEGIN
          DELETE(alfa,LENGTH(alfa),1);
          WRITE(alfa,' ');
          GOTOXY(x_pos,y_pos);
          WRITE (alfa);
        END;
   END;
 END
```

En el caso de variables de tipo de cadena utilizamos el procedimiento DELETE, que permite, en nuestro caso, borrar el último carácter de la cadena. La instrucción está constituida por tres parámetros, de los cuales el primero indica en qué cadena se quiere trabajar, el segundo indica qué carácter de la cadena se quiere borrar y el último parámetro sirve para indicar cuántos caracteres deben borrarse. En nuestro caso, la función LENGTH(ALFA) resti-

tuye la longitud de la variable ALFA, que sirve también para indicar la posición del último carácter.

```
ELSE BEGIN
 IF sw  IN ['1','2']
    THEN BEGIN
    IF car IN['0'..'9']
      THEN BEGIN
           IF sw='1'
            THEN pequeno:=pequeno*10+ORD(car)-48
            ELSE grande:=grande*10+ORD(car)-48;
            write
          END
      ELSE WRITE(CHR$(7));
         END  {del campo numerico}
```

Estas instrucciones se refieren al caso en que lo que queremos es introducir dígitos. Primero se comprueba que el carácter está comprendido entre los valores 0 y 9; de no ser así se emite una señal acústica. Si el carácter es, efectivamente, un dígito se añadirá a la variable PEQUEÑO o a la variable GRANDE, dependiendo del contenido de la variable SW y, luego, el carácter aparecerá en la pantalla.

```
ELSE BEGIN
      IF car IN [' '..'"']
        THEN BEGIN
             st:=' ';
             st[1]:=car;
             alfa:=CONCAT(alfa,st);
             WRITE(car);
            END
            ELSE WRITE(CHR$(7));
    END;   {del campo alfabetico}
```

Esta examina el caso de datos alfabéticos. Si el contenido de la variable CAR no está comprendido entre espacio y la letra "z" se emitirá una señal acústica y, de no ser así, se pondrá "a la cola" en la variable ALFA (ver función CONCAT) y el carácter aparecerá en la pantalla.

Subrutina menú

El operador tiene la posibilidad de "elegir" la función que desea respondiendo con un valor numérico. Las instrucciones de BASIC se llevan a los programas GESTION del apéndice, desde la instrucción 800 a la instrucción 980.

```
830 REM
840 PRINT "1 --> CARGA DE DATOS"
850 PRINT
860 PRINT "2 --> LECTURA DE REGISTROS"
870 PRINT
900 PRINT "4 --> MODIFICACION DE UN REGISTRO"
910 PRINT
920 PRINT "5 --> BORRADO DE UN REGISTRO"
930 PRINT
940 PRINT "9 --> FINAL DE PROGRAMA"
950 PRINT
960 PRINT "?":RESPUESTA$=INPUT$(1):QUEQUIERE=
    ASC(RESPUESTA$)-ASC("0")
```

La instrucción 850 tiene como objeto insertar una línea vacía entre una opción y otra del menú. La respuesta dada mediante el teclado se recoge en la variable RESPUESTA$, que luego se introduce en la variable QUEQUIERE en forma binaria. La función INPUT$ nos permite pulsar una tecla y volver a entrar en el programa sin tener que pulsar la tecla RETURN

En el apéndice reservado al Pascal se incluye el procedimiento MENU introducido en el programa GESTION. Veamos las instrucciones principales:

```
borrado_pantalla;
WRITELN;
WRITELN('1 --> CARGA DE DATOS');
WRITELN;
WRITELN('2 --> LECTURA CON EL NOMBRE DE CLIENTE');
WRITELN;
WRITELN('3 --> LECTURA CON EL NUMERO DE REGISTRO');
WRITELN;
WRITELN('4 --> MODIFICACION O ANULACION DE REGISTRO');
WRITELN;
WRITELN('5 --> PUESTA EN COLA DEL REGISTRO AL FINAL');
WRITELN;
```

```
WRITELN('9 --> FIN DEL PROGRAMA');
WRITELN;
READ(QUE_QUIERE);WRITELN;
```

La primera instrucción tiene como objeto borrar la pantalla; luego se visualizan las opciones del menú separadas por una línea vacía. La respuesta dada a través del teclado se recoge en la variable QUE_QUIERE. Se ha hecho de tal modo que no tuviéramos necesidad de utilizar la tecla RETURN; es suficiente pulsar una tecla cualquiera para volver el control al programa.

Subrutina de máscara

Esta subrutina escribe en la pantalla los nombres de los campos, dejando a su derecha el espacio adecuado para la respuesta; dicho de otro modo, se hace una "entrada guiada".

En BASIC la subrutina se lleva al apéndice desde la instrucción 10000 a la 10230. Comentemos algún fragmento de esta subrutina para ilustrar su funcionamiento.

```
10040 YPOS=4:XPOS=1
10050 GOSUB 15800:REM -->POSICIONA EL CURSOR
10060 PRINT "CLIENTE    :"
```

El cursor se posiciona de forma adecuada (línea 1, columna 4) y luego se imprime la cadena "CLIENTE :". A continuación siguen otras instrucciones, similares a las arriba indicadas, para presentar en pantalla la solicitud de la dirección, de la ciudad, del teléfono y del prefijo; por supuesto, en posiciones diferentes.

En el lenguaje PASCAL, el procedimiento se lleva al apéndice reservado al Pascal con el nombre MASCARA.

```
GOTOXY(0,4);
WRITE('cliente    :');
```

GOTOXY posiciona el cursor en la primera columna de la línea 5 y WRITE escribe el encabezamiento del campo del cliente. Siguen luego otras instrucciones, similares a las dos arriba indicadas, para describir los otros campos del registro.

Subrutina de visualización

En BASIC la subrutina ocupa las instrucciones 11000/11260 y se lleva íntegramente al apéndice para cada estructura.


```
11040 YPOS=4:XPOS=13
11050 GOSUB 15800:REM -->POSICIONA EL CURSOR
11060 PRINT CL$;
```

El cursor se posiciona en la columna 13 de la línea 4, en donde se imprime el nombre del cliente. Se procede de forma análoga para los otros campos del registro.

En PASCAL, el procedimiento VISUALIZACION se lleva al apéndice reservado al Pascal. Comentamos solamente algunas instrucciones:

```
GOTOXY(13,4);
WRITE(cliente^.cliente);
```

El cursor se posiciona en la columna 14 de la línea 5, en donde se imprime el nombre del cliente. Se procede de forma análoga para los otros campos del registro.

Subrutina de carga

Para el BASIC la subrutina se lleva al apéndice en los programas GESTION, a partir de la instrucción 2000 y hasta la instrucción 2740. Procedemos a comentar las instrucciones principales, recordando que los comentarios se refieren a la gestión HASH; durante el comentario de las otras estructuras indicaremos las ocasionales diferencias.

```
2040 GOSUB 500:REM -->BORRADO DE PANTALLA
2050 PRINT "LA CARGA PARTE CON EL NUMERO:";
     ACTUAL%-1
```

Después de haber borrado la pantalla se visualiza el número del primer registro a grabar.

```
2060 IF ACTUAL%<3 THEN GOTO 2140
2070 NUMERO%=ACTUAL%-1
2080 FIELD 1,31 AS CL$,33 AS IN$,15 AS CI$,
     4 AS TE$,4 AS PR$,1 AS TI$
2090 GOSUB 15000:REM -->LECTURA DE CLIENTE
2100 PRINT "EL REGISTRO ESCRITO ES EL SIGUIENTE:"
2110 GOSUB 10000:REM -->MASCARA
2120 GOSUB 11000:REM -->VISUALIZACION
2130 PRINT
```

```
2140 INPUT "PULSE <RETURN> PARA INICIAR LA CARGA"; RESPUESTA$
```

En el caso de que no se hubiera escrito todavía el primer registro, el control se pasará a la instrucción 2140. De no ser así (cuando ya se inició la carga) se leerá el último registro grabado y se le visualizará en la pantalla, por lo que se podrá comprobar la posición exacta de la carga. Se entra luego en un bucle de instrucciones en donde se piden los datos del cliente; dicho bucle termina cuando se responde con "zzzz" a la solicitud del nombre del cliente.

```
2150 REM
2155 FIELD 1,31 AS CL$,33 AS IN$,15 AS CI$,
     4 AS TE$,4 AS PR$,1 AS TI$
2160 GOSUB 500:REM ---->BORRADO DE PANTALLA
2170 YPOS=20:XPOS=1
2180 GOSUB 15800:REM -->POSICIONA EL CURSOR
2190 PRINT "PARA ACABAR LA CARGA,A LA SOLICITUD
     NOMBRE DEL CLIENTE TECLEE 'zzzz'"
2200 GOSUB 10000:REM -->MASCARA
2210 XPOS=13:YPOS=4
2220 SW=3
2230 GOSUB 1000:REM --->TECLADO
2240 NOMINATIVO$=ALFA$
2250 IF NOMINATIVO$="zzzz" THEN RETURN
```

El cursor se posiciona en la columna 13 de la línea 4 y luego se cede el control a la subrutina TECLADO. El valor restituido se almacena en la variable NOMINATIVO$ en el caso de que se introduzca "zzzz". En caso contrario:

```
2260 IF ACTUAL%>ULTIMO% THEN INPUT "NO PUEDE
     CONTINUAR.SE HA SUPERADO LA DIMENSION DEL FICHERO
```

Lo anterior sirve para cerciorarse de no haber superado las dimensiones del fichero.

```
2270 GOSUB 12000:REM --->CALCULO
2280 GOSUB 13000:REM --->BUSQUEDA
2290 IF FIN=1 GOTO 2320
2230 INPUT "ESTE CLIENTE YA EXISTE. NO PUEDO
     INTRODUCIRLO.PULSE <RETURN>";RESPUESTA$
2310 RETURN
```


La subrutina BUSQUEDA realiza la búsqueda en el fichero CLIENTES del nombre correspondiente y pone a 1 la variable FIN si dicho nombre está introducido ya en el fichero, en cuyo caso lo rechazaremos.

```
2320 CLIENTE$=NOMINATIVO$
2330 SW=3
2340 XPOS=13
2350 YPOS=6
2360 GOSUB 1000:REM --->TECLADO
2370 DIRECCION$=ALFA$
```

A través del teclado se introducen los datos correspondientes al campo de la dirección. Luego se solicitan de forma análoga las informaciones relativas a los otros campos.

```
2540 GOSUB 15200:REM --->LECTURA HASH
2550 COLISION%=CVI(HASH$)
```

Se lee el registro actual del fichero de índice y el valor del único campo se almacena en la variable COLISION%.

```
2560 LSET HASH$=MKI$(ACTUAL%)
2570 GOSUB 15700:REM ---> ESCRITURA HASH
```

Se escribe el registro de índice después de haberlo actualizado con el contenido de la variable ACTUAL%.

```
2580 FIELD 1,31 AS CL$, 33 AS IN$,15 AS CI$,
     4 AS TE$, 4 AS PR$,1 AS TI$, 2 AS CO$
2590 LSET CL$=CLIENTE$
```

Ahora actualizamos el fichero de clientes; con la última instrucción se transfiere el contenido de la variable CLIENTE$ a la variable CL$; destaquemos el hecho de que ha de utilizarse la instrucción LSET.

A continuación se dan otras instrucciones análogas para completar los demás campos del registro.

```
2650 LSET CO$=MKI$(COLISION%)
2660 NUMERO%=ACTUAL%
2670 GOSUB 15500:REM -->ESCRITURA CLIENTE
2680 ACTUAL%=ACTUAL%+1
```

Damos ahora las instrucciones correspondientes a la escritura del registro de los clientes y a la actualización del "cuenta registros".

```
2690 FIELD 1,2 AS ACTUAL$,2 AS ULTIMO$
2700 LSET ACTUAL$=MKI$(ACTUAL%)
2710 LSET ULTIMO$=MKI$(ULTIMO%)
2720 NUMERO%=1
2730 GOSUB 15500:REM -->ESCRITURA CLIENTE/1
2740 GOTO 2150
```

Se vuelve a escribir, actualizado, el primer registro del fichero de clientes para recordar el siguiente registro a grabar y el total de los registros del fichero. A continuación, el control se pasa a la instrucción 2150, en donde se solicitan otros datos del teclado.

El procedimiento CARGA se lleva al apéndice de Pascal y se incluye en los programas GESTION. Pasamos a comentar las instrucciones principales.

```
borrado_pantalla;
WRITELN('LA CARGA PARTE CON EL NUMERO',actual-1);
```

Después de haber borrado la pantalla, se visualiza el número del primer registro a grabar.

```
IF actual>2
 THEN BEGIN
  SEEK(cliente,PRED(actual)) ;
  GET(cliente);
  WRITELN('EL REGISTRO ANTERIOR',
          'ESCRITO ES EL SIGUIENTE')
  mascara;
  visualizacion;
  WRITELN;
  GOTOXY(0,20);
  WRITE('PULSE <RETURN> PARA CONTINUAR');
END;
```

En el caso de que se esté en fase avanzada de carga se lee el último registro grabado y se le visualiza en la pantalla, de modo que se pueda comprobar la posición.

```
REPEAT
 borrado_pantalla;
```



```
GOTOXY(0,20);
WRITELN('PARA TERMINAR RESPONDA "zzzz"',
        'A LA SOLICITUD DEL NOMBRE DEL CLIENTE');
mascara;
sw:='3';
x_pos:=13;
y_pos:=4;
teclado;
nominativo:=alfa;
IF nominativo<>'zzzz'
 THEN BEGIN
   IF actual>ultimo
    THEN BEGIN
     WRITE('NO PUEDO CONTINUAR,SE HA SUPERADO',
           'LA DIMENSION MAXIMA.PULSE<RETURN>');
     READLN;
     EXIT(carga);
   END;
```

Mediante REPEAT se inicia un bucle de instrucciones que se terminará cuando la variable NOMINATIVO contenga el valor "zzzz". El cursor se sitúa en la columna 31 de la línea 4 y luego el control se cede al procedimiento TECLADO. El valor restituido se almacena en la variable NOMINATIVO.

Si no hemos terminado la carga comprobamos que no hemos superado las dimensiones previstas del fichero, en cuyo caso obligamos a la salida de la subrutina mediante la instrucción EXIT.

```
calculo;
busqueda;
IF NOT fin
 THEN BEGIN
  WRITE('ESTE CLIENTE YA EXISTE, NO PUEDO',
        'INCLUIRLO. PULSE <RETURN>');
  READLN;
  EXIT(carga);
END;
```

El procedimiento BUSQUEDA realiza la búsqueda del nombre correspondiente en el fichero de clientes y en el caso de que lo encuentre pone a 1 (VERDADERO) la variable booleana FIN. Rechazaremos la introducción de claves que existan ya en el fichero.

```
cliente:=nominativo;
x_pos:=13;
y_pos:=6
teclado;
direccion:=alfa;
```

A través del procedimiento TECLADO se introducen los datos correspondientes al campo de dirección. Luego se solicitan, de forma análoga, los otros campos.

```
SEEK(indice,hash);
GET(indice)
colision:=indice^
```

Se lee el registro actual del fichero de índice y el valor del único campo se almacena en la variable COLISION.

```
indice^:=actual;
SEEK(indice,hash);
PUT(indice);
```

Se escribe el registro de índice después de haberlo actualizado con el contenido de la variable ACTUAL.

```
SEEK(cliente,actual);
PUT(cliente);
actual:=succ(actual);
```

Ahora actualizamos el fichero de clientes y luego hacemos lo mismo con el "cuenta-registros".

```
primero:=actual;
maximo:=ultimo;
SEEK(cliente,1);
PUT(CLIENTE);
```

Se actualiza y vuelve a escribir el registro 1 del fichero de clientes para recordar el siguiente registro a grabar y el total de los registros existentes en el fichero.

```
    END; {DE LA WITH CLIENTE^}
  END; {SI CLIENTE}
UNTIL nominativo='zzzz'
```


El bucle se repetirá a no ser que a la solicitud del nombre del cliente se responda con "zzzz", en cuyo caso se sale del bucle y del procedimiento.

Subrutina de lectura

Permite leer los registros y visualizarlos en la pantalla; solicita el número del primero y del último registro. La subrutina LISTA tiene el objeto de buscar el enésimo registro, siempre que se haya grabado; si el registro no existe, se activará la variable FIN. Se sale de la subrutina pulsando la tecla <ESCAPE> cuando se ha visualizado el último registro.

La subrutina se lleva al apéndice en los programas GESTION y ocupa las instrucciones 4000 a 4330. Veamos las instrucciones principales.

```
4040 YPOS=20:XPOS=1
4050 GOSUB 15800:REM -->POSICIONA EL CURSOR
4060 PRINT "NUMERO DEL PRIMER REGISTRO:"
```

Después de haber posicionado el cursor se pide el número del primer registro a visualizar; dicho número se incrementa en 1 y se almacena en la variable INICIO%. De forma análoga se pide el número del último registro a grabar y este valor, incrementado en 1, se almacena en la variable ALT%.

```
4210 GOSUB 11700:REM -->POSICIONA
4220 IF FINE=1 THEN RETURN
```

El control se pasa a la subrutina LISTA, que se encarga de buscar el primer registro; si no lo encuentra activará la variable FIN y se saldrá de la subrutina. De no ser así:

```
4240 GOSUB 500:REM ---->BORRADO DE PANTALLA
4250 GOSUB 10000:REM -->MASCARA
4260 GOSUB 11000:REM -->VISUALIZACION
```

El registro se visualiza en la pantalla. Siguen instrucciones que permiten observar con comodidad la pantalla y luego suspender el examen de los demás registros o visualizarlos.

El procedimiento LECTURA se lleva a los programas GESTION del apéndice de Pascal. Comentemos ahora las instrucciones principales:

```
PROCEDURE lectura
BEGIN
GOTOXY(0,20);
WRITE('NUMERO DEL PRIMER REGISTRO:     ');
x_pos:=32;
y_pos:=20;
sw:='1';
teclado;
inicio:=pequeno+1;
```

A través del teclado se solicita el número del primer registro a imprimir y dicho valor, incrementado en 1, se almacena en la variable INICIO. De forma análoga, se pide el número del último registro a visualizar y dicho valor, incrementado en 1, se almacena en la variable STOP.

```
lista;
IF NOT fin
THEN REPEAT
```

El control se pasa al procedimiento LISTA, que, si no se encuentra el registro, pone a 1 (VERDADERO) la variable booleana FIN. Si se encuentra el registro:

```
borrado_pantalla;
mascara;
visualizacion;
GOTOXY(0,20);
lista;
WRITE('<ESPACIO> PARA CONTINUAR,',
     '<ESC> PARA ACABAR');
READ(respuesta);
```

Se visualiza el registro, se lee el registro siguiente y se da la posibilidad al operador de observar los datos en la pantalla y proseguir luego con la visión de los demás registros.

```
UNTIL (respuesta=CHR$(27)) OR (inicio>alt+1) OR (fin);
```

La visualización de los registros termina:

- cuando el operador pulsa la tecla <ESCAPE>;
- cuando se ha visualizado el último registro solicitado;
- cuando se acaban los registros existentes en el fichero.

Subrutina de modificación

La subrutina de BASIC se lleva a los programas GESTION del apéndice BASIC, desde la instrucción 8000 a la instrucción 8760. Comentamos las instrucciones principales, recordando que los comentarios se refieren a la gestión HASH; durante el examen de las otras estructuras, indicaremos las ocasionales diferencias.

```
8070 PRINT "NUMERO DE REGISTRO:"
8080 SW=1
8090 XPOS=23:YPOS=20
8100 GOSUB 1000:REM --->TECLADO
8110 INICIO%=PEQUENO%+1
8120 GOSUB 11700:REM -->POSICIONA
8130 IF FIN=0 THEN GOTO 8210
8170 YPOS=20:XPOS=1
8180 GOSUB 15800:REM -->POSICIONA EL CURSOR
8190 INPUT "NO EXISTE ESTE REGISTRO,PULSE
     <RETURN>";RESPUESTA$
8200 RETURN
```

Se pide el nombre del cliente a través del teclado y la subrutina de cálculo determina el número Hash, mientras que la subrutina de búsqueda encuentra el registro. En el caso de que el nombre no se encuentre, se emitirá una señal acústica; si se encuentra:

```
8220 GOSUB 10000:REM -->MASCARA
8230 GOSUB 11000:REM -->VISUALIZACION
8240 YPOS=20:XPOS=1
8250 GOSUB 15800:REM -->POSICIONA EL CURSOR
8260 PRINT "PULSE <ESC> PARA MODIFICAR EL CAMPO
     O <RETURN> PARA CONTINUAR"
```

El registro se visualiza en la pantalla y el cursor se posiciona al comienzo de cada campo individual. El operador puede hacer dos cosas:

- pulsar la tecla RETURN: el cursor se situará en el campo siguiente;
- pulsar la tecla ESCAPE: entonces será posible volver a escribir el campo en donde se encuentra el cursor.

Las instrucciones principales relativas a la modificación del campo de dirección son:

```
8275 YPOS=6:XPOS=11
8280 GOSUB 15800:REM -->POSICIONA EL CURSOR
8290 GOSUB 15900:REM -->LECTURA CARACTER
8300 IF ASC(CAR$)<>27 THEN GOTO 8380
8310 YPOS=6:XPOS=13
8320 GOSUB 15800:REM -->POSICIONA EL CURSOR
8330 PRINT "
     "
8340 SW=3
8350 XPOS=13:YPOS=6
8360 GOSUB 1000:REM --->TECLADO
8370 LSET IN$=ALFA$
```

Las instrucciones correspondientes a los otros campos son muy similares, aunque, por supuesto, se cambia el posicionamiento del cursor. Al final, la subrutina sigue con:

```
8750 GOSUB 15500:REM -->ESCRITURA CLIENTE
8760 RETURN
```

En el apéndice de Pascal hemos llevado el procedimiento MODIFICACION al programa GESTION. Veamos las instrucciones principales:

```
WRITE('NUMERO DEL REGISTRO:');
sw:'1';
x_pos:=18;
y_pos:=20;
teclado;
inicio:=pequeno+1;
j:=pequeno+1;
lista;
IF NOT fin
THEN BEGIN
```

Se pide el registro a través del teclado Si no se encuentra se emitirá una señal acústica, si se localiza:

```
mascara;
visualizacion;
GOTOXY(0,20);
WRITE('<ESC> PARA MODIFICAR,',
      '<RETURN> PARA CONTINUAR');
```


El registro se visualiza en la pantalla, el cursor se posiciona al comienzo de cada campo individual y el operador puede hacer dos cosas:

- pulsar la tecla RETURN: el cursor se posicionará sobre el campo siguiente;
- pulsar la tecla ESCAPE: entonces será posible volver a escribir el campo en donde está situado el cursor.

Veamos las instrucciones principales para la modificación del campo de dirección:

```
GOTOXY(13,6);
READ(respuesta);
IF respuesta=CHR$(27)
 THEN BEGIN
  GOTOXY(13,6);
  WRITE('                    ');
  sw:='3';
  x_pos:=13;
  y_pos:=6;
  teclado;
  cliente^.direccion:=alfa;
```

Las instrucciones correspondientes a los otros campos son similares a las anteriores y cambiará el posicionamiento del cursor. Al final del procedimiento sigue:

```
SEEK(cliente,j);
PUT(cliente);
```

Subrutina de borrado

Llevamos la subrutina a los programas GESTION en el apéndice del BASIC (instrucciones 9000/9280). Comentaremos solamente las instrucciones principales, recordando que los comentarios se refiren a la gestion HASH y que durante el comentario de las otras estructuras indicaremos las eventuales diferencias.

```
9070 SW=1:XPOS=23:YPOS=20
9080 GOSUB 1000:REM --->TECLADO
9090 INICIO%=PEQUEÑO%+1
9110 GOSUB 11700:REM -->TECLADO
9120 IF FIN=1 THEN GOTO 9250
```

Solicita el nombre del cliente a borrar, luego el control se pasa a la subrutina CALCULO para la determinación del número Hash y, finalmente, a la subrutina BUSQUEDA para encontrar el registro. En el caso de que no se encuentre el nombre se activará la variable FIN y se indicará dicha circunstancia en la pantalla. Si se localiza:

```
9130 GOSUB 10000:REM -->MASCARA
9140 GOSUB 11000:REM -->VISUALIZACION
9150 YPOS=20:XPOS=1
9160 GOSUB 15800:REM -->POSICIONA EL CURSOR
9170 PRINT "¿QUIERE BORRARLO?(S/N)";
9180 GOSUB 15900:REM -->LECTURA CARACTER
```

El registro se visualiza en la pantalla y luego, habida cuenta del peligro de la operación, se pide la confirmación del borrado.

```
9190 IF CAR$="S" OR CAR$="s" THEN GOTO 9210
```

Si no se responde de forma afirmativa (con "S" o "s") se sale de la subrutina sin borrar el registro; en otro caso:

```
9220 LSET TI$="C"
9230 GOSUB 15500:REM -->ESCRITURA CLIENTE
9240 RETURN
```

El campo timpo se modifica en C y luego se graba el registro.

En el apéndice de Pascal y en los programas GESTION se lleva el procedimiento BORRADO del que comentamos las instrucciones siguientes:

```
WRITE('NUMERO DEL REGISTRO:  ');
sw:='1';
x_pos:=18;
y_pos:=20;
teclado;
inicio:=pequeno+1;
j:=pequeno+1;
lista;
IF NOT fin
THEN BEGIN
```

Se pide el nombre del registro a borrar. En el caso de que no se encuentre la variable booleana FIN se pone a 1 (VERDADERO) de modo que se pueda indicar dicha circunstancia en la pantalla. En caso contrario:

```
mascara;
visualizacion;
GOTOXY(0,20);
WRITE('¿QUIERE BORRARLO?(S/N)');
READ(respuesta);
```

El registro se visualiza en la pantalla y luego, dada la delicadeza de la operación, se solicita la confirmación correspondiente.

```
IF respuesta IN['S','s']
 THEN BEGIN
   cliente^.tipo:='C';
   SEEK(cliente,j);
   PUT(cliente);
 END;
```

Se modifica el campo tipo y luego se graba el registro.

FICHEROS SECUENCIALES

Sabemos que los ficheros secuenciales nos permiten el acceso a un registro a condición de que se hayan procesado los eventuales registros anteriores. Además no siempre es posible añadir otros registros a un fichero que se haya cerrado con anterioridad. Algunas versiones de BASIC han previsto una opción particular en la apertura del fichero (la opción APPEND) que tiene, precisamente, el objeto de añadir nuevos registros "a la cola". El PASCAL UCSD no prevé dicha técnica.

Los programas se han elaborado de modo que sigan lo más posible las reglas de la programación estructurada. Ello fue fácil para el Pascal y un poco más difícil con el BASIC, aunque hemos utilizado las instrucciones REM. Nos vimos obligados a utilizar un Pascal muy simple para poderlo transformar "de igual a igual" en BASIC; dicho de otro modo: no hemos pasado parámetros al cierre de los procedimientos.

Hemos empleado registros estructurados de la forma siguiente:

CLIENTE	campo alfabético de 31 caracteres
DIRECCION	campo alfabético de 33 caracteres

CIUDAD campo alfabético de 15 caracteres
TELEFONO campo numérico de 8 caracteres
PREFIJO campo alfabético de 4 caracteres

Programa SEC_GESTION

El programa está subdividido en numerosas subrutinas y el cuerpo está reducido a lo más fundamental; se presenta un menú y basándose en la respuesta dada se elige una de las cinco funciones básicas: carga de datos, lectura de datos, modificación/borrado de datos y adición de registros "a la cola".

En BASIC el programa se lleva al apéndice con el nombre de SEC_GESTION y ocupa las instrucciones 220 a 290.

```
220 GOSUB 500:REM -->BORRADO DE PANTALLA
230 GOSUB 800:REM -->MENU
240 ON QUEQUIERE GOSUB 2000,4000,6000,8000,9000
250 IF QUEQUIERE<>9 THEN GOTO 220
290 END
```

Las dos primeras instrucciones tienen su propia explicación, porque hemos utilizado instrucciones REM. La instrucción 240 cede el control a la instrucción 2000, si la variable QUEQUIERE contiene el valor 1; a la instrucción 4000, si la variable QUEQUIERE contiene el valor 2, y así sucesivamente. Si la variable QUEQUIERE contiene valores superiores al 5, el control pasará a la instrucción 220 si este valor es diferente de 9 y, de no ser así, el programa se cerrará.

Las instrucciones de Pascal relativas al programa se llevan al apéndice en el programa SEC_GESTION y, como es habitual, comentamos algunas instrucciones:

```
BEGIN
  menu;
  CASE que_quiere OF
        '1':          carga;
        '2':         nombre;
        '3':        lectura;
        '4':   modificacion;
        '5':  puesta_en_fila;
  END;
END.
```


Nos parece que está bastante claro, ¿no?

Veamos ahora las diversas subrutinas siguiendo una cierta lógica, que es la de examinar antes las relativas a la carga de los datos, después las de lectura, luego las de modificación y borrado y finalmente las de "puesta a la cola".

Subrutina de lectura

Permite leer los registros a partir del enésimo y hasta otro registro determinado.

En BASIC la subrutina se lleva al apéndice en el programa SEC_GESTION, desde la instrucción 4000 a la instrucción 4370. Como es habitual, comentamos las instrucciones más importantes.

```
4035 OPEN "I",#1,"CLIENTES.DAT"
```

Con dicha instrucción se abre, solamente como entrada, el fichero CLIENTES.DAT.

```
4060 PRINT "NUMERO DEL PRIMER REGISTRO:"
4070 SW=1
4080 XPOS=32
4090 YPOS=20
4100 GOSUB 1000: REM -->TECLADO
4110 INICIO%=PEQUEÑO%+1
```

Se solicita el número del primer registro a imprimir y dicho valor se almacena en la variable INICIO%. Más adelante, en las instrucciones 4120/4190, se solicita el número del último registro a imprimir y dicho valor se almacena en la variable ALT%.

```
4210 GOSUB 11700:REM ->POSICIONA
4220 IF FIN=1 THEN THEN GOTO 4340
```

La subrutina POSICIONA busca el primer registro a imprimir y, en el caso de que no se encuentre, activará la variable FIN y sale de la subrutina. De no ser así:

```
4240 GOSUB 500:REM --->BORRADO DE PANTALLA
4250 GOSUB 10000:REM ->MASCARA
4260 GOSUB 11000:REM ->VISUALIZACION
4270 YPOS=20:XPOS=1
4280 GOSUB 15800:REM -->POSICIONA EL CURSOR
```

```
4290 GOSUB 11500:REM -->LISTA
4300 PRINT "<ESPACIO> PARA SEGUIR,<ESC> PARA
     TERMINAR"
4310 RESPUESTA$=INPUT$(1)
```

En la pantalla se visualiza todo el registro y en la última línea se escribe un mensaje y se espera la respuesta. De este modo se tiene la posibilidad de observar la pantalla y luego decidir si desea ver los siguientes registros o terminar; si pulsa la barra espaciadora se visualizarán los registros sucesivos (si existen) y si se pulsa la tecla <ESCAPE> se suspende la visualización de los registros.

```
4320 IF ASC(RESPUESTA)=27 OR FIN=1 OR INICIO%>
     ALT%+1 THEN GOTO 4340
4330 GOTO 4230
```

La impresión de los registros queda suspendida en los casos siguientes:

- si el operador ha pulsado la tecla ESCAPE,
- si no hay otros registros en el fichero (EOF),
- si se ha realizado la visualización del último registro indicado por el operador.

En otro caso se prosigue la visualización de los demás registros, cediendo el control a la instrucción 4240.

El procedimiento LECTURA se lleva al apéndice reservado al Pascal, en el programa SEC_GESTION. Comentamos las instrucciones principales:

```
WRITE('NUMERO DEL PRIMER REGISTRO:   ');
sw:='1';
x_pos:=32;
y_pos:=20;
teclado;
inicio:=pequeno;
```

Se solicita el número del primer registro a visualizar; el valor correspondiente se almacena en la variable INICIO. A continuación se solicita el número del último registsro a imprimir, cuyo valor se almacena en la variable ALT.

```
posiciona;
IF NOT fin
 THEN REPEAT
```



```
   borrado_pantalla;
   mascara;
   visualizacion;
   GOTOXY(0,20);
   GET(cliente);
   inicio:=inicio+1;
   WRITE('<ESPACIO> PARA SEGUIR,',
         '<ESC> PARA TERMINAR');
   READ(respuesta);
  UNTIL  (respuesta=CHR$(27))
        OR (inicio>alt)
        OR (EOF(cliente));
```

La subrutina POSICIONA busca el primer registro a visualizar; si dicho registro no existe se activará la variable booleana FIN y se sale de la subrutina. De no ser así, el registro se visualizará en la pantalla. Se imprime un mensaje y se espera la respuesta para dar la posibilidad al operador de observar a su voluntad y luego decidir si seguir con la visualización de los demás registros (pulsa la barra espaciadora) o si interrumpir la subrutina (pulsa la tecla de escape). La visualización de los demás registros se suspenderá en el caso de que:

- el operador haya pulsado la tecla <ESCAPE>,
- no haya otros registros en el fichero (EOF),
- se haya visualizado el último registro indicado por el operador.

Subrutina POSICIONA

Esta subrutina lee registros, a partir del primero, hasta que encuentra el deseado. En el caso de que no encuentre el registro dará la indicación oportuna a través de la variable FIN.

En el apéndice de BASIC, la subrutina está incluida en el programa SEC_GESTION, desde la instrucción 11700 a la instrucción 11850.

```
11750 NUMERO%=NUMERO%+1
11760 IF NUMERO%>INICIO% THEN 11830
11770 GOSUB 15000:REM -->LECTURA CLIENTE
11780 IF EOF(1) THEN GOTO 11830
11790 IF NUMERO%=INICIO% THEN GOTO 11810
```

```
11800 GOTO 11740
11810 FIN=0
11820 RETURN
11830 FIN=1
```

Se llama a la subrutina de lectura del cliente hasta que:

- se haya encontrado el registro correspondiente;
- se haya llegado al final del fichero, en cuyo caso se pone la variable FIN a 1;
- NUMERO% sea superior a INICIO%, en cuyo caso se pone la variable FIN a 1.

En Pascal, el procedimiento se lleva al apéndice en el programa SEC_GESTION.

```
x:=0;
REPEAT
 x:=x+1;
 IF x<inicio then GET (cliente);
UNTIL (x=inicio)
       OR EOF(cliente);
     fin:=EOF(cliente);
```

Se seguirán leyendo los registros del fichero de clientes hasta que el valor de X corresponda al valor de INICIO o bien se haya llegado al final del fichero.

Subrutina de carga

Para el BASIC la subrutina se lleva al apéndice a partir de la instrucción 2000 y hasta la instrucción 2720. Procedamos a comentar las instrucciones principales:

```
2040 OPEN "O",1,"CLIENTES.DAT"
```

Hemos abierto el fichero; luego introduciremos los datos de nuestros clientes. También en el caso de la carga de datos tendremos que llegar a su final. Hemos establecido que al introducir desde el teclado el valor "zzzz" se pretende acabar la carga, considerando como muy improbable que haya un cliente con dicho apellido.


```
2060 GOSUB 13500:REM -->INTRODUCCION DATOS
2070 IF NOMINATIVO$="zzzz" THEN GOTO 2700
2590 CL$=CLIENTE$
2600 IN$=DIRECCION$
2610 CI$=CIUDAD$
2620 TE$=ALFA$
2630 PR$=PREFIJO$
2670 GOSUB 15500:REM -->ESCRITURA CLIENTE
2680 GOTO 2050
2700 CLOSE 1
2710 RETURN
```

Las subrutinas llamadas en las instrucciones 2680 y 2670 se comentarán a continuacaión y el comentario se adjuntará con la indicación de para qué sirven. Desde la instrucción 2590 a la instrucción 2630, según puede observarse en el apéndice, los valores introducidos a través del teclado se transfieren a los campos del registro. Si a la solicitud del nombre del cliente se responde con "zzzz", el control se cede a la instrucción 2700, en donde se cierra el fichero y luego se entra de nuevo en la instrucción que efectúa la llamada.

En Pascal utilizaremos el procedimiento CARGA, que se lleva al apéndice precisamente en el programa SEC_GESTION. Dicho procedimiento es muy sencilli, pero, no obstante, comentaremos algunas instrucciones.

```
REWRITE(cliente,'cliente.dat');
REPEAT
solicitudes;
IF nominativo<>'zzzz' THEN PUT(cliente);
UNTIL nominativo='zzzz';
```

Tenga presente que el control se cede al procedimiento SOLICITUDES, que se encarga de requerir información a través del teclado; la subrutina correspondiente se comentará más adelante. Si a la solicitud del nombre del cliente se responde con "zzzz" se saldrá de la subrutina CARGA y se cierra el fichero; de no ser así se grabará el registro y se volverán a pedir datos.

Subrutina SOLICITUDES

Esta subrutina borra la pantalla, presenta la máscara de solicitud de datos (en la práctica, los nombres de los campos) y pide

los datos correspondientes al nombre, a la dirección, a la ciudad, al número de teléfono y al prefijo correspondiente. Hace uso de la subrutina TECLADO, ya comentada, para comprobar tanto los datos alfabéticos como los numéricos.

La subrutina BASIC se lleva al apéndice desde la línea 13500 a la 13870 del programa SEC_GESTION. Como es nuestra costumbre, comentaremos las instrucciones principales:

```
13550 GOSUB 500 ---->BORRADO DE PANTALLA
13560 YPOS=20:XPOS=1
13570 GOSUB 15800:REM -->POSICIONA EL CURSOR
13580 PRINT "PARA TERMINAR,A LA SOLICITUD DEL
      NOMBRE DEL CLIENTE RESPONDA 'zzzz'"
13590 GOSUB 10000:REM -->MASCARA
13600 XPOS=13:YPOS=4
13610 SW=3
13620 GOSUB 1000:REM --->TECLADO
13630 NOMINATIVO$=ALFA$
13640 IF NOMINATIVO$="zzzz" THEN RETURN
13650 CLIENTE$=NOMINATIVO$
```

Borramos la pantalla y posicionamos adecuadamente el cursor, de modo que la subrutina MASCARA visualice en la pantalla la explicación de lo que debe introducirse a través del teclado. La instrucción 13620 cede el control a la subrutina TECLADO y puesto que hemos establecido en tres el valor de la variable SW se deberán suministrar datos alfabéticos. Solamente después de haber pulsado RETURN se produce el reenvío de la subrutina del teclado y se ejecuta la instrucción 13630; posteriormente se comprueba si se ha introducido el valor "zzzz", en cuyo caso se termina la carga; en caso contrario, el contenido de la variable ALFA$ se transfiere a la variable CLIENTE$.

```
13660 SW=3
13670 XPOS=13
13680 YPOS=6
13690 GOSUB 1000:REM --->TECLADO
13700 DIRECCION$=ALFA$
```

Con estas instrucciones hemos introducido desde el teclado la dirección del cliente. A continuación hay otras instrucciones, muy similares a éstas, para introducir los otros campos del registro.

En Pascal el procedimiento SOLICITUDES se introduce en el programa SEC_GESTION que se lleva al apéndice. Procedamos a comentar algunas instrucciones:

```
WRITELN('PARA FINALIZAR LA CARGA,RESPONDA',
        '"zzzz" A LA PETICION DE CLIENTE');
mascara;
x_pos:=13;
y_pos:=4;
sw:='3';
teclado;
nominativo:=alfa;
IF nominativo<>'zzzz'
 THEN BEGIN
 WITH cliente^
 DO BEGIN
    cliente:=nominativo;
```

De modo similar se solicitan la dirección, la ciudad, el teléfono y el prefijo.

Subrutina LISTA

Sigue la lectura del registro y, en el caso de que se verifique el final del fichero, pone a 1 la variable FIN.

La subrutina BASIC está contenida en el apéndice en el programa SEC_GESTION desde la instrucción 11500 a la instrucción 11620.

En PASCAL no se activa ningún procedimiento porque las instrucciones están comprendidas ya en el procedimiento LECTURA.

Subrutina NOMBRE

Muestra el contenido de los registros a partir de un nombre de cliente hasta otro nombre de cliente, indicados ambos por el usuario. Después de la visualización del registro siempre será posible renunciar a la visualización de los sucesivos registros con la pulsación de la tecla <ESCAPE>.

En BASIC, la subrutina está contenida en el apéndice en el programa SEC_GESTION, instrucciones 6000/6410. Como es nuestra costumbre, comentamos las instrucciones principales:

```
6040 OPEN "i",1,"clientes.dat"
```

El fichero CLIENTES.DAT se abre en entrada, es decir en Input.

```
6090 PRINT "NOMBRE DEL PRIMER CLIENTE:"
6100 SW=3
6110 XPOS=32
6120 YPOS=20
6130 GOSUB 1000:REM --->TECLADO
6140 POSICION$=ALFA$
```

Se solicita el nombre del primer cliente a imprimir, que se almacena en la variable PARTIDA$. De manera similar se solicita el nombre del último cliente a imprimir y dicho nombre se almacena en la variable TERMINO$.

```
6240 GOSUB 13000:REM -->CLAVE
6250 IF FIN=1 THEN GOTO 6380
```

La subrutina CLAVE, que se comentará dentro de poco, tiene el cometido de buscar el nombre del primer cliente; si no lo encuentra, activará la variable FIN y se saldrá de la subrutina NOMBRE.

```
6260 GOSUB 500:REM ---->BORRADO DE PANTALLA
6270 GOSUB 10000:REM -->MASCARA
6280 GOSUB 11000:REM -->VISUALIZACION
6290 YPOS=20:XPOS=1
6300 GOSUB 15800:REM -->POSICIONA EL CURSOR
6310 PRINT "<ESPACIO> PARA SEGUIR,<ESC> PARA
     CONTINUAR"
```

En la pantalla se visualiza el registro y en la última línea aparecerá un mensaje con espera de respuesta. De este modo se puede observar con comodidad la presentación visual y luego, pulsando la barra espaciadora, se tiene la posibilidad de ver los registros sucesivos y pulsando la tecla <ESCAPE> se suspende la visualización de los registros.

```
6320 GOSUB 15900:REM -->LECTURA CARACTER
6330 IF ASC(CAR$)=27 THEN GOTO 6380
6340 IF TERMINO$=CL$ THEN GOTO 6380
6350 IF EOF(1) THEN GOTO 6380
6360 GOSUB 15000:REM -->LECTURA CLIENTE
6370 GOTO 6260
```


Se sale de la subrutina:

- si se pulsa la tecla <ESCAPE>;
- si el nombre del cliente visualizado corresponde al último nombre que se quería imprimir;
- si no hay otros registros en el fichero.

De no ser así, se lee un nuevo registro y se cede el control a la instrucción 6260.

En Pascal, el procedimiento NOMBRE se lleva al apéndice en el programa SEC_GESTION. El fichero CLIENTES.DAT se abre como entrada y luego se solicita el nombre del primer cliente a leer, almacenándolo en PARTIDA; finalmente se solicita el nombre del último cliente almacenándolo en la variable TERMINO.

```
nominativo:=partida;
clave;
IF NOT fin
 THEN REPEAT
  borrado_pantalla;
  mascara;
  visualizacion;
  fuera:=cliente^.cliente=termino;
  GOTOXY(0,20);
  GET(cliente);
  WRITE('<ESPACIO> PARA SEGUIR,',
        '<ESC> PARA TERMINAR');
  READ(respuesta);
 UNTIL (respuesta)=CHR$(27)
       OR (fuera)
       OR EOF(cliente);
```

El procedimiento CLAVE se encarga de encontrar el primer nombre. Si no lo encuentra pone a 1 (VERDADERO) la variable booleana FIN. Si encuentra el primer nombre activará un bucle de instrucciones controlado por REPEAT UNTIL. Con dichas instrucciones se visualiza en la pantalla el registro y se da la posibilidad al operador de visualizar los registros sucesivos o de interrumpir el bucle (pulsando la tecla <ESCAPE>). El bucle se interrumpirá:

- si el operador pulsa la tecla <ESCAPE>;
- si se ha visualizado el último nombre deseado;
- si se han acabado los registros del fichero (EOF).

Es interesante la instrucción FUERA := CLIENTES^.CLIENTE=TERMINO, en donde el nombre del cliente se compara con el contenido de la variable TERMINO; si los dos valores son diferentes, FUERA toma el valor de FALSO y si los dos valores son iguales, FUERA toma el valor de VERDADERO.

Subrutina CLAVE

Esta subrutina examina los registros desde el inicio del fichero y busca el nombre del cliente que corresponde al contenido de la variable NOMINATIVO. En el caso de que el cliente no se encuentre se activará la variable FIN.

En BASIC la subrutina se lleva al apéndice, desde la línea 13000 a la 13080 del programa SEC_GESTION. Consideramos que dicha subrutina está suficientemente documentada.

En Pascal, el procedimiento CLAVE se lleva al apéndice en el programa SEC_GESTION. El núcleo del procedimiento consiste en un bucle de instrucciones que comienza con una instrucción REPEAT y termina con la instrucción UNTIL(CLIENTE=NOMINATIVO) OR EOF(CLIENTES). La variable FIN se pone a cero (FALSO) si hay todavía registros en el fichero, y, de no ser así, se pone a 1 (VERDADERO).

Subrutina MODIFICACION

Sirve tanto para modificar como para borrar registros. En los ficheros secuenciales no es posible modificar el estado de un fichero si no se crea otro nuevo que comprenda las modificaciones. El nuevo fichero tiene como nombre externo CLIENTES.NOV. Al comienzo de la subrutina se abren los ficheros, el primero en lectura (entrada) y el segundo en escritura (salida, es decir, se trata de un fichero nuevo).

Más adelante se leen todos los registros del fichero de clientes. Cada registro se presenta en la pantalla y el operador tiene la posibilidad de dejarlo intacto, modificarlo o suprimirlo. Si el registro debe volver a copiarse se escribirá en el fichero de salida (output). Si el registro debe suprimirse no se escribe en el nuevo fichero, pero antes de ejecutar esta función se solicita la confirmación correspondiente al operador. Si el registro ha de modificarse es posible cambiar cualquier campo. El cursor se posiciona al comienzo del campo, y si el operador quiere modificarlo, pulsará la tecla <ESCAPE>, luego escribirá los nuevos valores, concluyendo con la tecla <RETURN>. Si no quiere modificar el campo, el operador pulsará la tecla <RETURN>.

En BASIC la subrutina se lleva al apéndice y ocupa las instrucciones 8000/8830 del programa SEC_GESTION. Procedemos a comentar las instrucciones principales:

```
8040 OPEN "I",1,"CLIENTES.DAT"
8050 OPEN "O",2,"CLIENTES.NOV"
```

El fichero CLIENTES.DAT se abre como entrada (Input), mientras se crea un nuevo fichero, de nombre CLIENTE.NOV, que contendrá la nueva situación de clientes.

```
8070 IF EOF(1) THEN GOTO 8730
8080 GOSUB 15000:REM -->LECTURA CLIENTE
8090 GOSUB 500:REM ---->BORRADO DE PANTALLA
8100 REM
8110 GOSUB 10000:REM -->MASCARA
8120 GOSUB 11000:REM -->VISUALIZACION
8130 YPOS=20:XPOS=1
8140 GOSUB 15800:REM -->POSICIONA EL CURSOR
8150 PRINT "M)ODIFICA,B)ORRA,R)ECOPIA";
```

Cada registro se presenta en la pantalla; podrá copiarse en el nuevo fichero si se da una respuesta "R", se modificará si se responde "M" y, finalmente, se suprimirá si se responde "B". Si se responde con caracteres diferentes a los indicados, se repetirá la solicitud.

Si quiere modificar el registro se ejecutarán las instrucciones 8210 a 8270. En la práctica se ejecutan instrucciones similares para cada campo del registro. Por ejemplo, para poder modificar el campo de la dirección del cliente se hace:

```
8230 YPOS=6:XPOS=11
8240 GOSUB 15800:REM -->POSICIONA EL CURSOR
8250 GOSUB 15900:REM -->LECTURA CARACTER
8260 IF ASC(CAR$)<>27 THEN GOTO 8340
```

Hemos colocado el cursor junto al campo de la dirección y si el operador responde con la tecla <ESC> se proseguirá con las instrucciones siguientes; de no ser así, continúa con el examen del campo siguiente:

```
8270 YPOS=6:XPOS=13
8280 GOSUB 15800:REM -->POSICIONA EL CURSOR
8290 PRINT "
     "
```

El cursor se ha posicionado en el campo a modificar y la anterior situación del campo se elimina de la pantalla.

```
8300 SW=3
8310 XPOS=13:YPOS=6
8320 GOSUB 1000:REM --->TECLADO
8330 IN$=ALFA$
```

La subrutina TECLADO restituye en la variable ALFA$ la nueva dirección, que se transfiere a la variable IN$. Después de haber visualizado los demás campos, el registro se grabará y luego se vuelven a leer y visualizar otros registros con el empleo de las dos instrucciones siguientes:

```
8710 WRITE #2,CL$,IN$,CI$,TE$,PR$
8720 GOTO 8070
```

Si el registro se recopia tal como está en el nuevo fichero se ejecutarán las instrucciones desde 8700 a 8720. Si se anula el registro se ejecutan las instrucciones desde 8770 hasta 8820.

```
8780 GOSUB 15800:REM -->POSICIONA EL CURSOR
8790 PRINT "oDESEA BORRARLO? (S/N)";
8800 GOSUB 15900:REM -->LECTURA CATACTER
8810 IF CAR$="s" OR "S" THEN GOTO 8070
8820 GOTO 8700
```

En el caso de que se responda con "S" o "s" no se graba el registro, sino que se lee el siguiente del fichero de entrada; si no se responde de forma afirmativa, el registro se grabará sin ninguna modificación. Cuando acabemos los registros del fichero de entrada (Input) se ejecutan las instrucciones siguientes:

```
8730 CLOSE 1
8740 CLOSE 2
8750 RETURN
```

Al apéndice de Pascal llevamos el procedimiento MODIFICACION, introducido en el programa SEC_GESTION. Procedamos a comentar las instrucciones principales:

```
RESET  (cliente,'cliente.dat');
REWRITE(actualizacion,'cliente.nov');
```


Se abre el fichero CLIENTES.DAT como entrada, mientras que como salida (Output) se abre el nuevo fichero CLIENTES.NOV. Las instrucciones sucesivas se repetirán varias veces hasta que no se hayan acabado los registros del fichero de entrada.

```
borrado_pantalla;
mascara;
visualizacion;
GOTOXY(0,20);
WRITE('¿M)odificacion,C)ancelar,R)ecopia?:');
READ(respuesta);
```

Cada registro se visualizará en la pantalla y el operador podrá:

- recopiarlo en el nuevo fichero, si responde con R;
- borrarlo, si responde con B;
- modificarlo, si responde con M.

En el caso de que no se responda de manera adecuada, se volverá a hacer la solicitud correspondiente. Si se quiere borrar el registro, se solicitará la confirmación al operador, y si este último da una respuesta negativa, el registro se llevará al fichero ACTUALIZACION.

APENDICE A

APENDICE BASIC

ACCESO ALEATORIO

PREPARACION

```
100 REM ****************************************
110 REM *           RAN_PREPARA                *
120 REM ****************************************
130 REM
140 FICHERO$="CLIENTES.DAT"
160 OPEN "R",1,FICHERO$,88
170 INPUT "¿DE CUANTOS REGISTROS ESTARA COMPUESTO
    EL FICHERO?:";ULTIMO
180 PRIMERO=2
190 FIELD 1,2 AS PRIMERO$,2 AS ULTIMO$
200 LSET PRIMERO$=MKI$(PRIMERO)
210 LSET ULTIMO$=MKI$(ULTIMO+1)
220 PUT #1,1
230 FIELD 1,31 AS CLIENTE$,33 AS DIRECCION$,15 AS
    CIUDAD$,4 AS TELEFONO$,4 AS PREFIJO$,1 AS TIPO$
250 LSET CLIENTE$=""
260 LSET DIRECCION$=""
270 LSET CIUDAD$=""
280 LSET TELEFONO$="0"
290 LSET PREFIJO$="0000"
300 LSET TIPO$=" "
320 FOR X%=2 TO ULTIMO+1
```

```
330 PUT #1,X%
340 NEXT X%
350 CLOSE #1
400 END
```

ACCESO ALEATORIO

GESTION

```
100 REM ##################################
110 REM #           RAN_GESTION          #
120 REM ##################################
130 REM
140 OPEN "R",1,"CLIENTES.DAT",88
170 NUMERO%=1
180 FIELD 1,2 AS ACTUAL$,2 AS ULTIMO$
190 GOSUB 15000:REM ---->LEE CLIENTE/1
200 ACTUAL%=CVI(ACTUAL$)
210 ULTIMO%=CVI(ULTIMO$)
220 GOSUB 500:REM ------>BORRADO DE PANTALLA
230 GOSUB 800:REM ------>MENU
240 ON QUEQUIERE GOSUB 2000,4000,6000,8000,9000
250 IF QUEQUIERE <>9 THEN GOTO 220
270 CLOSE 1
290 END
500 REM ::::::::::::::::::::::::::
510 REM ::  BORRADO DE PANTALLA  ::
520 REM ::::::::::::::::::::::::::
530 REM
540 CLS
550 RETURN
800 REM ::::::::::::
810 REM ::  MENU  ::
820 REM ::::::::::::
830 REM
840 PRINT "1 --> CARGA DE DATOS"
850 PRINT
860 PRINT "2 --> LECTURA DE REGISTROS"
870 PRINT
```



```
900 PRINT "4 --> MODIFICACION DE UN REGISTRO"
910 PRINT
920 PRINT "5 --> BORRADO DE UN REGISTRO"
930 PRINT
940 PRINT "9 --> FINAL DE PROGRAMA"
950 PRINT
960 PRINT "?":RESPUESTA$=INPUT$(1):QUEQUIERE=
    ASC(RESPUESTA$)-ASC("0")
980 RETURN
1000 REM ::::::::::::::
1010 REM ::  TECLADO  ::
1020 REM ::::::::::::::
1030 REM
1040 ALFA$=""
1050 GOSUB 15800:REM --->POSICIONA EL CURSOR
1060 GOSUB 15900:REM --->LECTURA DEL CARACTER
1070 IF ASC(CAR$)=13 THEN GOTO 1210
1080 IF ASC(CAR$)<>8 THEN GOTO 1310
1090 REM -----------------------
1100 REM --  TECLA DE RETORNO --
1110 REM -----------------------
1120 IF LEN(ALFA$)>1 THEN ALFA$=LEFT$(ALFA$,
     LEN (ALFA$)-1) ELSE ALFA$=""
1130 REM ---------------------------
1140 REM --  IMPRESION DEL CAMPO  --
1150 REM ---------------------------
1160 GOSUB 15800:REM -->POSICIONA EL CURSOR
1170 PRINT ALFA$;" ";
1180 GOSUB 15800:REM -->POSICIONA EL CURSOR
1190 PRINT ALFA$
1200 GOTO 1060
1210 REM ::::::::::::::::::::
1220 REM ::  NUEVA ENTRADA  ::
1230 REM ::::::::::::::::::::
1240 IF SW=3 THEN RETURN
1250 G#=0
1260 FOR X=1 TO LEN(ALFA$)
1270 G#=G#*10+ASC(MID$(ALFA$,X,1))-48
1280 NEXT X
1290 IF SW=1 THEN PEQUEÑO$=CSNG(G#) ELSE
     GRANDE#=G#
```

```
1300 RETURN
1310 REM -----------------------
1320 REM --  CARACTER NORMAL  --
1330 REM -----------------------
1340 IF SW=3 THEN GOTO 1450
1350 REM -------------------------
1360 REM --  CARACTER NUMERICO  --
1370 REM -------------------------
1380 IF CAR$<"0" OR CAR$>"9" THEN GOTO 1400
1390 GOTO 1490
1400 REM ----------------------
1410 REM --  ERROR DE TECLA  --
1420 REM ----------------------
1430 PRINT CHR$(7);
1440 GOTO 1060
1450 REM ---------------------------
1460 REM --  CARACTER ALFABETICO  --
1470 REM ---------------------------
1480 IF CAR$<" " OR CAR$>"z" THEN GOTO 1400
1490 ALFA$=ALFA$+CAR$
1500 GOTO 1130
2000 REM :::::::::::::::::::::
2010 REM ::  CARGA DE DATOS  ::
2020 REM :::::::::::::::::::::
2030 REM
2040 GOSUB 500:REM -->BORRADO DE PANTALLA
2050 PRINT "LA CARGA PARTE CON EL NUMERO:"; ACTUAL%-1
2060 IF ACTUAL%<3 THEN GOTO 2140
2070 NUMERO%=ACTUAL%-1
2080 FIELD 1,31 AS CL$,33 AS IN$,15 AS CI$,
     4 AS TE$,4 AS PR$,1 AS TI$
2090 GOSUB 15000:REM -->LECTURA DE CLIENTE
2100 PRINT "EL REGISTRO ESCRITO ES EL SIGUIENTE:"
2110 GOSUB 10000:REM -->MASCARA
2120 GOSUB 11000:REM -->VISUALIZACION
2130 PRINT
2140 INPUT "PULSE <RETURN> PARA INICIAR LA CARGA";
     RESPUESTA$
2150 REM
2155 FIELD 1,31 AS CL$,33 AS IN$,15 AS CI$,
     4 AS TE$,4 AS PR$,1 AS TI$
```



```
2160 GOSUB 500:REM ---->BORRADO DE PANTALLA
2170 YPOS=20:XPOS=1
2180 GOSUB 15800:REM -->POSICIONA EL CURSOR
2190 PRINT "PARA ACABAR LA CARGA,A LA SOLICITUD
     NOMBRE DEL CLIENTE TECLEE 'zzzz'"
2200 GOSUB 10000:REM -->MASCARA
2210 XPOS=13:YPOS=4
2220 SW=3
2230 GOSUB 1000:REM --->TECLADO
2240 NOMINATIVO$=ALFA$
2250 IF NOMINATIVO$="zzzz" THEN RETURN
2260 IF ACTUAL%>ULTIMO% THEN INPUT "NO PUEDE
     CONTINUAR.SE HA SUPERADO LA DIMENSION DEL
     FICHERO
2270 GOSUB 12000:REM --->CALCULO
2280 GOSUB 13000:REM --->BUSQUEDA
2290 IF FIN=1 GOTO 2320
2230 INPUT "ESTE CLIENTE YA EXISTE. NO PUEDO
     INTRODUCIRLO.PULSE <RETURN>";RESPUESTA$
2310 RETURN
2320 CLIENTE$=NOMINATIVO$
2330 SW=3
2340 XPOS=13
2350 YPOS=6
2360 GOSUB 1000:REM --->TECLADO
2370 DIRECCION$=ALFA$
2380 SW=3
2390 XPOS=13
2400 YPOS=8
2410 GOSUB 1000:REM --->TECLADO
2420 CIUDAD$=ALFA
2430 SW=2
2440 XPOS=13
2450 YPOS=10
2460 GOSUB 1000:REM --->TECLADO
2470 TELEFONO#=GRANDE#
2480 SW=3
2490 S=XPOS=13
2500 YPOS=12
2510 GOSUB 1000:REM --->TECLADO
2520 PREFIJO$=ALFA$
```

```
2530 TIPO$=" "
2540 GOSUB 15200:REM --->LECTURA HASH
2550 COLISION%=CVI(HASH$)
2560 LSET HASH$=MKI$(ACTUAL%)
2570 GOSUB 15700:REM ---> ESCRITURA HASH
2580 FIELD 1,31 AS CL$, 33 AS IN$,15 AS CI$,
     4 AS TE$, 4 AS PR$,1 AS TI$, 2 AS CO$
2590 LSET CL$=CLIENTE$
2600 LSET IN$=DIRECCION$
2610 LSTE CI$=CIUDAD$
2620 LSET TE$=MKI$(TELEFONO#)
2630 LSET PR$=PREFIJO$
2640 LSET TI$=TIPO$
2650 LSET CO$=MKI$(COLISION%)
2660 NUMERO%=ACTUAL%
2670 GOSUB 15500:REM -->ESCRITURA CLIENTE
2680 ACTUAL%=ACTUAL%+1
2690 FIELD 1,2 AS ACTUAL$,2 AS ULTIMO$
2700 LSET ACTUAL$=MKI$(ACTUAL%)
2710 LSET ULTIMO$=MKI$(ULTIMO%)
2720 NUMERO%=1
2730 GOSUB 15500:REM -->ESCRITURA CLIENTE/1
2740 GOTO 2150
2750 REM
4000 REM :::::::::::::::::::::::::::::::::::::::
4010 REM ::  LECTURA CON EL NUMERO DE REGISTRO  ::
4020 REM :::::::::::::::::::::::::::::::::::::::
4030 REM
4040 YPOS=20:XPOS=1
4050 GOSUB 15800:REM -->POSICIONA EL CURSOR
4060 PRINT "NUMERO DEL PRIMER REGISTRO:"
4070 SW=1
4080 XPOS=32
4090 YPOS=20
4100 GOSUB 1000:REM --->TECLADO
4110 INICIA%=PEQUEÑO%+1
4120 YPOS=20:XPOS=1
4130 GOSUB 15800:REM -->POSICIONA CURSOR
4140 PRINT "NUMERO DEL ULTIMO REGISTRO:"
4150 SW=1
4160 XPOS=32
```



```
4170YPOS=20
4180 GOSUB 1000:REM --->TECLADO
4190 ALT%=PEQUEÑO%+1
4200 IF ALT%>ACTUAL-1 THEN ALT%=ACTUAL%-1
4210 GOSUB 11700:REM -->POSICIONA
4220 IF FINE=1 THEN RETURN
4230 REM
4240 GOSUB 500:REM ---->BORRADO DE PANTALLA
4250 GOSUB 10000:REM -->MASCARA
4260 GOSUB 11000:REM -->VISUALIZACION
4270 YPOS=20:XPOS=1
4280 GOSUB 15800:REM -->POSICIONA EL CURSOR
4290 GOSUB 11500:REM -->LISTA
4300 PRINT "<ESPACIO> PARA SEGUIR,<ESCAPE> PARA
     ACABAR"
4310 RESPUESTA$=INPUT$(1)
4320 IF ASC(RESPUESTA$)=27 OR FIN=1 OR INICIO%>
     ALT%+1 THEN RETURN
4330 GOTO 4230

6000 REM ::::::::::::::::::::::
6010 REM ::  ERROR DE TECLADO  ::
6020 REM ::::::::::::::::::::::
6030 REM
6040 RETURN
6050 REM

8000 REM ::::::::::::::::::
8010 REM ::  MODIFICACION  ::
8020 REM ::::::::::::::::::
8030 REM
8040 GOSUB 500:REM ---->BORRADO DE PANTALLA
8050 YPOS=20:XPOS=1
8060 GOSUB 15800:REM -->POSICIONA EL CURSOR
8070 PRINT "NUMERO DE REGISTRO:"
8080 SW=1
8090 XPOS=23:YPOS=20
8100 GOSUB 1000:REM --->TECLADO
8110 INICIO%=PEQUEÑO%+1
8120 GOSUB 11700:REM -->POSICIONA
8130 IF FIN=0 THEN GOTO 8210
```

```
8170 YPOS=20:XPOS=1
8180 GOSUB 15800:REM -->POSICIONA EL CURSOR
8190 INPUT "NO EXISTE ESTE REGISTRO,PULSE
     <RETURN>";RESPUESTA$
8200 RETURN
8210 REM
8220 GOSUB 10000:REM -->MASCARA
8230 GOSUB 11000:REM -->VISUALIZACION
8240 YPOS=20:XPOS=1
8250 GOSUB 15800:REM -->POSICIONA EL CURSOR
8260 PRINT "PULSE <ESC> PARA MODIFICAR EL CAMPO
     O <RETURN> PARA CONTINUAR"
8261 YPOS=4:XPOS=11
8262 GOSUB 15800:REM -->POSICIONA EL CURSOR
8263 GOSUB 15900:REM -->LECTURA CARACTER
8264 IF ASC(CAR$)<>27 THEN GOTO 8275
8265 YPOS=4:XPOS=13
8266 GOSUB 15800:REM -->POSICIONA EL CURSOR
8267 PRINT "
     "
8268 SW=3:YPOS=4:XPOS=13
8269 GOSUB 1000:REM --->TECLADO
8270 LSET CL$=ALFA$
8275 YPOS=6:XPOS=11
8280 GOSUB 15800:REM -->POSICIONA EL CURSOR
8290 GOSUB 15900:REM -->LECTURA CARACTER
8300 IF ASC(CAR$)<>27 THEN GOTO 8380
8310 YPOS=6:XPOS=13
8320 GOSUB 15800:REM -->POSICIONA EL CURSOR
8330 PRINT "
     "
8340 SW=3
8350 XPOS=13:YPOS=6
8360 GOSUB 1000:REM --->TECLADO
8370 LSET IN$=ALFA$
8380 REM
8390 YPOS=8:XPOS=11
8400 GOSUB 15800:REM -->POSICIONA EL CURSOR
8410 GOSUB 15900:REM -->LECTURA CARACTER
8420 IF ASC(CAR$)<>27 THEN GOTO 8500
8430 YPOS=8:XPOS=13
```



```
8440 GOSUB 15800:REM -->POSICIONA EL CURSOR
8450 PRINT "
     "
8460 SW=3
8470 GOSUB 15800:REM -->POSICIONA EL CURSOR
8480 GOSUB 1000:REM --->TECLADO
8490 LSET CI$=ALFA$
8500 REM
8510 YPOS=10:XPOS=11
8520 GOSUB 15800:REM -->POSICIONA EL CURSOR
8530 GOSUB 15900:REM -->LECTURA CARACTER
8540 IF ASC(CAR$)<>27 THEN GOTO 8620
8550 YPOS=10:XPOS=13
8560 GOSUB 15800:REM POSICIONA EL CURSOR
8570 PRINT "
     "
8580 SW=2
8590 XPOS=13:YPOS=10
8600 GOSUB 1000:REM --->TECLADO
8610 LSET TE$=MKS$(GRANDE$#)
8620 REM
8630 YPOS=12:XPOS=11
8640 GOSUB 15800:REM -->POSICIONA EL CURSOR
8650 GOSUB 15900:REM -->LECTURA CARACTER
8660 IF ASC(CAR$)<>27 THEN GOTO 8740
8670 YPOS=12:XPOS=13
8680 GOSUB 15800:REM -->POSICIONA EL CURSOR
8690 PRINT "
     "
8700 SW=3
8710 YPOS=12:XPOS=13
8720 GOSUB 1000:REM --->TECLADO
8730 LSET PR$=ALFA$
8740 REM
8750 GOSUB 15500:REM -->ESCRITURA CLIENTE
8760 RETURN
8770 REM
9000 REM ::::::::::::::::
9010 REM ::  ANULACION  ::
9020 REM ::::::::::::::::
9030 REM
```

```
9040 GOSUB 500:REM ---->BORRADO DE PANTALLA
9050 YPOS=20:XPOS=1
9060 GOSUB 15800:REM -->POSICIONA EL CURSOR
9065 PRINT "NUMERO DEL REGISTRO:"
9070 SW=1:XPOS=23:YPOS=20
9080 GOSUB 1000:REM --->TECLADO
9090 INICIO%=PEQUEÑO%+1
9110 GOSUB 11700:REM -->TECLADO
9120 IF FIN=1 THEN GOTO 9250
9130 GOSUB 10000:REM -->MASCARA
9140 GOSUB 11000:REM -->VISUALIZACION
9150 YPOS=20:XPOS=1
9160 GOSUB 15800:REM -->POSICIONA EL CURSOR
9170 PRINT "¿QUIERE BORRARLO?(S/N)";
9180 GOSUB 15900:REM -->LECTURA CARACTER
9190 IF CAR$="S" OR CAR$="s" THEN GOTO 9210
9200 RETURN
9210 REM
9220 LSET TI$="C"
9230 GOSUB 15500:REM -->ESCRITURA CLIENTE
9240 RETURN
9250 REM
9260 YPOS=20:XPOS=1
9270 INPUT "NO EXISTE ESTE CLIENTE,PULSE
     <RETURN>";RESPUESTA$
9280 RETURN
10000 REM ::::::::::::::
10010 REM ::  MASCARA  ::
10020 REM ::::::::::::::
10030 REM
10040 YPOS=4:XPOS=1
10050 GOSUB 15800:REM -->POSICIONA EL CURSOR
10060 PRINT "CLIENTE    :"
10070 YPOS=6:XPOS=1
10080 GOSUB 15800:REM -->POSICIONA EL CURSOR
10090 PRINT "DIRECCION  :"
10100 YPOS=8:XPOS=1
10110 GOSUB 15800:REM -->POSICIONA EL CURSOR
10120 PRINT "CIUDAD     :"
10130 YPOS=10:XPOS=1
10140 GOSUB 15800:REM -->POSICIONA EL CURSOR
```



```
10150 PRINT "TELEFONO   :"
10160 YPOS=12:XPOS=1
10170 GOSUB 15800:REM -->POSICIONA EL CURSOR
10180 PRINT "PREFIJO    :"
10250 RETURN
10260 REM
11000 REM ::::::::::::::::::::
11010 REM ::  VISUALIZACION  ::
11020 REM ::::::::::::::::::::
11030 REM
11040 YPOS=4:XPOS=13
11050 GOSUB 15800:REM -->POSICIONA EL CURSOR
11060 PRINT CL$;
11070 IF TI$<>" " THEN PRINT "  ANULADO";
11080 YPOS=6:XPOS=13
11090 GOSUB 15800:REM -->POSICIONA EL CURSOR
11100 PRINT IN$
11110 YPOS=8:XPOS=13
11120 GOSUB 15800:REM -->POSICIONA EL CURSOR
11130 PRINT CI$
11140 YPOS=10:XPOS=13
11150 GOSUB 15800:REM -->POSICIONA EL CURSOR
11160 TELEFONO#=CVI$(TE$)
11170 PRINT TELEFONO#
11180 YPOS12:XPOS=13
11190 GOSUB 15800:REM -->POSICIONA EL CURSOR
11200 PRINT PR$
11290 RETURN
11300 REM
11500 REM :::::::::::
11510 REM :: LISTA  ::
11520 REM :::::::::::
11530 REM
11540 FIN=0
11550 IF INICIO%>=ACTUAL% OR INICIO%<=0 THEN
      GOTO 11610
11560 FIELD 1,31 AS CL$,33 AS IN$,15 AS CI$,
      4 AS TE$,4 AS PR$,1 AS TI$
11570 NUMERO%=INICIO%
11580 GOSUB 15000:REM -->LECTURA CLIENTES
11590 INICIO%=INICIO%+1
```

```
11600 RETURN
11610 FIN=1
11620 RETURN
11700 REM ::::::::::::::::
11710 REM ::  POSICIONA  ::
11720 REM ::::::::::::::::
11730 NUMERO%=0
11740 REM
11750 NUMERO%=NUMERO%+1
11760 IF NUMERO%>INICIO% THEN 11830
11770 GOSUB 15000:REM -->LECTURA CLIENTE
11780 IF EOF(1) THEN GOTO 11830
11790 IF NUMERO%=INICIO% THEN GOTO 11810
11800 GOTO 11740
11810 FIN=0
11820 RETURN
11830 FIN=1
11840 RETURN
11850 REM
15000 REM :::::::::::::::::::::::::::::::
15010 REM ::  LECTURA FICHERO CLIENTES  ::
15020 REM :::::::::::::::::::::::::::::::
15030 REM
15040 GET #1,NUMERO%
15050 RETURN
15090 REM
15500 REM :::::::::::::::::::::::::::::::::::
15510 REM ::  ESCRITURA FICHEROS CLIENTES  ::
15520 REM :::::::::::::::::::::::::::::::::::
15530 REM
15550 PUT #1,NUMERO%
15560 RETURN
15590 REM
15800 REM :::::::::::::::::::::::::
15810 REM ::  POSICIONA EL CURSOR  ::
15820 REM :::::::::::::::::::::::::
15830 REM
15840 LOCATE YPOS,XPOS,1
15850 RETURN
15860 REM
15900 REM ::::::::::::::::::::::::::::
```



```
15910 REM ::  LECTURA DE UN CARACTER  ::
15920 REM ::::::::::::::::::::::::::::
15930 REM
15940 CAR$=INPUT$(1)
15950 RETURN
15960 REM
```

ACCESO ALEATORIO

COMPACTACION

```
100 REM ******************************
110 REM **        RAN_COMPACT        **
120 REM ******************************
130 REM
140 OPEND "R",2,"CLIENTE.DAT",88
160 OPEND "R",1,"CLIENTE.VIE",88
170 FIELD 2,2 AS ACTUAL$,2 AS ULTIMO$
175 GET #2,1
180 FIELD 1,2 AS ATT$,2 AS ULT$
190 GET #1,1
210 X%=CVI$(ULTIMO$)
220 Y%=CVI$(ATT$)
230 NUMERO%=1
235 GOSUB 15500:REM -->ESCRITURA DE CLIENTE
240 ACTUAL%=2
250 Z%=2
260 REM
270 FIELD 1,31 AS CLI$,33 AS IND$,15 AS CIT$,
    4 AS TEL$,4 AS PRE$,1 AS TIP$
280 GET #1,Z%
290 IF TIP$<>"C" THEN GOTO 320
300 PRINT CLI$,"  ANULADO"
310 GOTO 475
320 REM
330 IF Z%>X% THEN GOTO 500
340 NOMINATIVO$=CLI$
345 FIELD 2,31 AS CL$,33 AS IN$,15 AS CI$,
    4 AS TE$,4 AS PR$,1 AS TI$
```

```
360 LSET CL$=CLI$
370 LSET IN$=IND$
380 LSET CI$=CIT$
390 LSET TE$=TEL$
400 LSET PR$=PRE$
410 LSET TI$=TIP$
450 PRINT CL$
460 NUMERO%=ACTUAL%
470 ACTUAL%=ACTUAL%+1
475 Z%=Z%+1
480 IF Z%<Y% THEN GOTO 260
490 GOTO 600
500 REM
510 PRINT "HAY ",Z%," REGISTROS ACTIVOS EN UN
    FICHERO QUE LOS PREVEE",X%,"!"
520 PRINT "NO PUEDO CONTINUAR"
530 INPUT "PULSE RETURN";RESPUESTA$
540 GOTO 650
600 REM
610 FIELD 2,2 AS ATT$,2 AS ULT$
620 NUMERO%=1
625 GOSUB 15000:REM --->LECTURA DE CLIENTES
630 LSET ATT$=MKI$(ACTUAL%)
640 GOSUB 15500:REM --->ESCRITURA DE CLIENTES
650 CLOSE 1
660 CLOSE 2
670 END
15000 REM ::::::::::::::::::::::::::::::::
15010 REM ::  LECTURA FICHERO CLIENTES  ::
15020 REM ::::::::::::::::::::::::::::::::
15030 REM
15040 GET #2,NUMERO%
15050 RETURN
15060 REM
15500 REM ::::::::::::::::::::::::::::::::::
15510 REM ::  ESCRITURA FICHERO CLIENTES  ::
15520 REM ::::::::::::::::::::::::::::::::::
15530 REM
15540 PUT #2,NUMERO%
15550 RETURN
15560 REM
```


GESTION SECUENCIAL EN BASIC

```
100 REM ******************
110 REM **  SEC_GESTION  **
120 REM ******************
130 REM
220 GOSUB 500:REM -->BORRADO DE PANTALLA
230 GOSUB 800:REM -->MENU
240 ON QUEQUIERE GOSUB 2000,4000,6000,8000,9000
250 IF QUEQUIERE<>9 THEN GOTO 220
290 END
500 REM ::::::::::::::::::::::::::
510 REM ::  BORRADO DE PANTALLA  ::
520 REM ::::::::::::::::::::::::::
530 REM
540 CLS
550 RETURN
800 REM :::::::::::
810 REM ::  MENU  ::
820 REM :::::::::::
830 REM
840 PRINT "1 --> CARGA DE DATOS"
850 PRINT
860 PRINT "2 --> LECTURA CON EL NUMERO DE REGISTROS"
870 PRINT
880 PRINT "3 --> LECTURA CON EL NOMBRE DE CLIENTE"
890 PRINT
900 PRINT "4 --> MODIFICACION O ANULACION DE REGISTRO"
910 PRINT
920 PRINT "5 --> PUESTA EN FILA DEL REGISTRO"
930 PRINT
940 PRINT "9 --> FIN DEL PROGRAMA"
950 PRINT
960 PRINT "?":RESPUESTA$=INPUT$(1):QUEQUIERE=
    ASC(RESPUESTA$)-ASC("0")
980 RETURN
1000 REM ::::::::::::::
1010 REM ::  TECLADO  ::
1020 REM ::::::::::::::
1030 REM
```

```
1040 ALFA$=""
1050 GOSUB 15800:REM -->POSICIONA EL CURSOR
1060 GOSUB 15900:REM -->LECTURA CARACTER
1070 IF ASC(CAR$)=13 THEN 1210
1080 IF ASC(CAR$)<>8 THEN 1310
1090 REM ------------------------
1100 REM --  TECLA DE RETORNO  --
1110 REM ------------------------
1120 IF LEN(ALFA$)>1 THEN ALFA$=LEFT$(ALFA$,
     LEN(ALFA$)-1) ELSE ALFA$=""
1130 REM --------------------------
1140 REM --  IMPRESION DE CAMPO  --
1150 REM --------------------------
1160 GOSUB 15800:REM -->POSICIONA EL CURSOR
1170 PRINT ALFA$;" ";
1180 GOSUB 15800:REM -->POSICIONA EL CURSOR
1190 PRINT ALFA$;
1200 GOTO 1060
1210 REM ---------------------
1220 REM --  NUEVA ENTRADA  --
1230 REM ---------------------
1240 IF SW=3 THEN RETURN
1250 G#=0
1260 FOR X=1 TO LEN(ALFA$)
1270 G#=G#*10+ASC(MID$(ALFA$,X,1))-48
1280 NEXT X
1290 IF SW=1 THEN PEQUEÑO=CSNG(G#) ELSE
     GRANDE#=G#
1300 RETURN
1310 REM -----------------------
1320 REM --  CARACTER NORMAL  --
1330 REM -----------------------
1340 IF SW=3 THEN GOTO 1450
1350 REM -------------------------
1360 REM --  CARACTER NUMERICO  --
1370 REM -------------------------
1380 IF CAR$<"0" OR CAR$>"9" THEN GOTO 1400
1390 GOTO 1490
1400 REM ----------------------
1410 REM --  ERROR DE TECLA  --
1420 REM ----------------------
```



```
1430 PRINT CHR$(7);
1440 GOTO 1060
1450 REM ---------------------------
1460 REM --  CARACTER ALFABETICO  --
1470 REM ---------------------------
1480 IF CAR$<" " OR CAR$<"z" THEN GOTO 1400
1490 ALFA$=ALFA$+CAR$
1500 GOTO 1130
2000 REM ::::::::::::::::::
2010 REM ::  CARGA DATOS  ::
2020 REM ::::::::::::::::::
2030 REM
2040 OPEN "O",1,"CLIENTES.DAT"
2050 REM
2060 GOSUB 13500:REM -->INTRODUCCION DATOS
2070 IF NOMINATIVO$="zzzz" THEN GOTO 2700
2590 CL$=CLIENTE$
2600 IN$=DIRECCION$
2610 CI$=CIUDAD$
2620 TE$=ALFA$
2630 PR$=PREFIJO$
2670 GOSUB 15500:REM -->ESCRITURA CLIENTE
2680 GOTO 2050
2700 CLOSE 1
2710 RETURN
2720 REM
4000 REM ::::::::::::::::::::::::::::::::::::::::
4010 REM ::  LECTURA CON EL NUMERO DE REGISTRO  ::
4020 REM ::::::::::::::::::::::::::::::::::::::::
4030 REM
4035 OPEN "I",#1,"CLIENTES.DAT"
4040 YPOS=20:XPOS=1
4050 GOSUB 15800:REM -->POSICIONA EL CURSOR
4060 PRINT "NUMERO DEL PRIMER REGISTRO:"
4070 SW=1
4080 XPOS=32
4090 YPOS=20
4100 GOSUB 1000: REM -->TECLADO
4110 INICIO%=PEQUEÑO%+1
4120 YPOS=20:XPOS=1
4130 GOSUB 15800:REM -->POSICIONA EL CURSOR
```

```
4140 PRINT "NUMERO DEL ULTIMO REGISTRO:
4150 SW=1
4160 XPOS=32
4170 YPOS=20
4180 GOSUB 1000:REM -->TECLADO
4190 ALT%=PEQUEÑO%+1
4210 GOSUB 11700:REM ->POSICIONA
4220 IF FIN=1 THEN THEN GOTO 4340
4230 REM
4240 GOSUB 500:REM --->BORRADO DE PANTALLA
4250 GOSUB 10000:REM ->MASCARA
4260 GOSUB 11000:REM ->VISUALIZACION
4270 YPOS=20:XPOS=1
4280 GOSUB 15800:REM -->POSICIONA EL CURSOR
4290 GOSUB 11500:REM -->LISTA
4300 PRINT "<ESPACIO> PARA SEGUIR,<ESC> PARA
     TERMINAR"
4310 RESPUESTA$=INPUT$(1)
4320 IF ASC(RESPUESTA)=27 OR FIN=1 OR INICIO%>
     ALT%+1 THEN GOTO 4340
4330 GOTO 4230
4340 REM
4350 CLOSE 1
4360 RETURN
4370 REM
6000 REM :::::::::::::
6010 REM ::  NOMBRE  ::
6020 REM :::::::::::::
6030 REM
6040 OPEN "i",1,"clientes.dat"
6050 FIN=0
6060 YPOS=20
6070 XPOS=1
6080 GOSUB 15800:REM -->POSICIONA EL CURSOR
6090 PRINT "NOMBRE DEL PRIMER CLIENTE:"
6100 SW=3
6110 XPOS=32
6120 YPOS=20
6130 GOSUB 1000:REM --->TECLADO
6140 POSICION$=ALFA$
6150 YPOS=20
```



```
6160 XPOS=1
6170 GOSUB 15800:REM -->POSICIONA EL CURSOR
6180 PRINT "NOMBRE DE ULTIMO CLIENTE:
         "
6190 SW=3
6200 XPOS=32
6210 YPOS=20
6220 GOSUB 1000:REM --->TECLADO
6230 TERMINO$=ALFA$
6240 GOSUB 13000:REM -->CLAVE
6250 IF FIN=1 THEN GOTO 6380
6260 GOSUB 500:REM ---->BORRADO DE PANTALLA
6270 GOSUB 10000:REM -->MASCARA
6280 GOSUB 11000:REM -->VISUALIZACION
6290 YPOS=20:XPOS=1
6300 GOSUB 15800:REM -->POSICIONA EL CURSOR
6310 PRINT "<ESPACIO> PARA SEGUIR,<ESC> PARA
     CONTINUAR"
6320 GOSUB 15900:REM -->LECTURA CARACTER
6330 IF ASC(CAR$)=27 THEN GOTO 6380
6340 IF TERMINO$=CL$ THEN GOTO 6380
6350 IF EOF(1) THEN GOTO 6380
6360 GOSUB 15000:REM -->LECTURA CLIENTE
6370 GOTO 6260
6380 REM
6390 CLOSE 1
6400 RETURN
6410 REM
8000 REM ::::::::::::::::::
8010 REM ::  MODIFICACION  ::
8020 REM ::::::::::::::::::
8030 REM
8040 OPEN "I",1,"CLIENTES.DAT"
8050 OPEN "O",2,"CLIENTES.NOV"
8060 REM
8070 IF EOF(1) THEN GOTO 8730
8080 GOSUB 15000:REM -->LECTURA CLIENTE
8090 GOSUB 500:REM ---->BORRADO DE PANTALLA
8100 REM
8110 GOSUB 10000:REM -->MASCARA
8120 GOSUB 11000:REM -->VISUALIZACION
```

```
8130 YPOS=20:XPOS=1
8140 GOSUB 15800:REM -->POSICIONA EL CURSOR
8150 PRINT "M)ODIFICA,B)ORRA,R)ECOPIA";
8160 GOSUB 15900:REM -->LECTURA CARACTER
8170 IF CAR$="C" OR CAR$="c" THEN GOTO 8770
8180 IF CAR$="M" OR CAR$="m" THEN GOTO 8210
8190 IF CAR$="R" OR CAR$="r" THEN GOTO 8700
8200 GOTO 8160
8210 YPOS=20:XPOS=1
8220 PRINT "PULSE <ESC> PARA MODIFICAR EL
     CAMPO,<RET> PARA CONTINUAR"
8230 YPOS=6:XPOS=11
8240 GOSUB 15800:REM -->POSICIONA EL CURSOR
8250 GOSUB 15900:REM -->LECTURA CARACTER
8260 IF ASC(CAR$)<>27 THEN GOTO 8340
8270 YPOS=6:XPOS=13
8280 GOSUB 15800:REM -->POSICIONA EL CURSOR
8290 PRINT "
     "
8300 SW=3
8310 XPOS=13:YPOS=6
8320 GOSUB 1000:REM --->TECLADO
8330 IN$=ALFA$
8340 REM
8350 YPOS=8:XPOS=11
8360 GOSUB 15800:REM -->POSICIONA EL CURSOR
8370 GOSUB 15900:REM -->LECTURA CARACTER
8380 IF ASC(CAR$)<>27 THEN GOTO 8460
8390 YPOS=8:XPOS=13
8400 GOSUB 15800:REM -->POSICIONA EL CURSOR
8410 PRINT "
     "
8420 SW=3
8430 GOSUB 15800:REM -->POSICIONA EL CURSOR
8440 GOSUB 1000:REM --->TECLADO
8450 CI$=ALFA$
8460 REM
8470 YPOS=10:XPOS=11
8480 GOSUB 15800:REM -->POSICIONA EL CURSOR
8490 GOSUB 15900:REM -->LECTURA CARACTER
8500 IF ASC(CAR$)<>27 THEN GOTO 8580
```



```
8510 YPOS=10:XPOS=13
8520 GOSUB 15800:REM -->POSICIONA EL CURSOR
8530 PRINT "
     "
8540 SW=2
8550 XPOS=13:YPOS=10
8560 GOSUB 1000:REM --->TECLADO
8570 TE$=ALFA$
8580 REM
8590 YPOS=12:XPOS=11
8600 GOSUB 15800:REM -->POSICIONA EL CURSOR
8610 GOSUB 15900:REM -->LECTURA CARACTER
8620 IF ASC(CAR$)<>27 THEN GOTO 8700
8630 YPOS=12:XPOS=13
8640 GOSUB 15800:REM -->POSICIONA EL CURSOR
8650 PRINT "
     "
8660 SW=3
8670 YPOS=12:XPOS=13
8680 GOSUB 1000:REM --->TECLADO
8690 PR$=ALFA$
8700 REM
8710 WRITE #2,CL$,IN$,CI$,TE$,PR$
8720 GOTO 8070
8730 CLOSE 1
8740 CLOSE 2
8750 RETURN
8760 REM
8770 YPOS=20:XPOS=1
8780 GOSUB 15800:REM -->POSICIONA EL CURSOR
8790 PRINT "oDESEA BORRARLO? (S/N)";
8800 GOSUB 15900:REM -->LECTURA CATACTER
8810 IF CAR$="s" OR "S" THEN GOTO 8070
8820 GOTO 8700
8830 REM
9000 REM :::::::::::::::::::::
9010 REM ::  PUESTA EN FILA  ::
9020 REM :::::::::::::::::::::
9030 REM
9040 OPEN "CLIENTES.DAT" FOR APPEND AS #1
9050 REM
```

```
9060 GOSUB 13500:REM -->INTRODUCCION DE DATOS
9070 IF NOMINATIVO$="zzzz" THEN GOTO 9200
9090 CL$=CLIENTE$
9100 IN$=DIRECCION$
9110 CI$=CIUDAD$
9120 TE$=ALFA$
9130 PR$=PREFIJO$
9140 GOSUB 15500:REM -->ESCRITURA CLIENTE
9150 GOTO 9050
9200 CLOSE 1
9210 RETURN
9220 REM
10000 REM ::::::::::::::
10010 REM ::  MASCARA  ::
10020 REM ::::::::::::::
10030 REM
10040 YPOS=4:XPOS=1
10050 GOSUB 15800:REM -->POSICIONA EL CURSOR
10060 PRINT "CLIENTE   :"
10070 YPOS=6:XPOS=1
10080 GOSUB 15800:REM -->POSICIONA EL CURSOR
10090 PRINT "DIRECCION :"
10100 YPOS=8:XPOS=1
10110 GOSUB 15800:REM -->POSICIONA EL CURSOR
10120 PRINT "CIUDAD    :"
10130 YPOS=10:XPOS=1
10140 GOSUB 15800:REM -->POSICIONA EL CURSOR
10150 PRINT "TELEFONO  :"
10160 YPOS=12:XPOS=1
10170 GOSUB 15800:REM -->POSICIONA EL CURSOR
10180 PRINT "PREFIJO   :"
10220 RETURN
12300 REM
11000 REM ::::::::::::::::::::
11010 REM ::  VISUALIZACION  ::
11020 REM ::::::::::::::::::::
11030 REM
11040 YPOS=6:XPOS=13
11050 GOSUB 15800:REM -->POSICIONA EL CURSOR
11060 PRINT CL$
11080 YPOS=6:XPOS=13
```



```
11090 GOSUB 15800:REM -->POSICIONA EL CURSOR
11100 PRINT IN$
11110 YPOS=8:XPOS=13
11120 GOSUB 15800:REM -->POSICIONA EL CURSOR
11130 PRINT CI$
11140 YPOS=10:XPOS=13
11150 GOSUB 15800:REM -->POSICIONA EL CURSOR
11170 PRINT TE$
11180 YPOS=12:XPOS=13
11190 GOSUB 15800:REM -->POSICIONA EL CURSOR
11200 PRINT PR$
11250 RETURN
11260 REM
11500 REM ::::::::::::
11510 REM ::  LISTA  ::
11520 REM ::::::::::::
11530 REM
11540 FIN=0
11550 IF EOF(1) THEN GOTO 11610
11580 GOSUB 15000:REM -->LECTURA CLIENTE
11590 INICIO%=INICIO%+1
11600 RETURN
11610 FIN=1
11620 RETURN
11700 REM ::::::::::::::::
11710 REM ::  POSICIONA  ::
11720 REM ::::::::::::::::
11730 REM
11740 NUMERO%=0
11750 NUMERO%=NUMERO%+1
11760 IF NUMERO%>INICIO% THEN 11830
11770 GOSUB 15000:REM -->LECTURA CLIENTE
11780 IF EOF(1) THEN GOTO 11830
11790 IF NUMERO%=INICIO% THEN GOTO 11810
11800 GOTO 11740
11810 FIN=0
11820 RETURN
11830 FIN=1
11840 RETURN
11850 REM
13000 REM ::::::::::::
```

```
13010 REM ::  CLAVE  ::
13020 REM ::::::::::::
13030 REM
13040 GOSUB 15000:REM -->LECTURA CLIENTE
13050 IF CL$=PARTIDA$ THEN RETURN
13060 IF EOF(1) THEN FIN=1:RETURN
13070 GOTO 13030
13080 REM
13500 REM
13510 REM ::::::::::::::::::
13520 REM :: SOLICITUDES  ::
13530 REM ::::::::::::::::::
13540 REM
13550 GOSUB 500 ---->BORRADO DE PANTALLA
13560 YPOS=20:XPOS=1
13570 GOSUB 15800:REM -->POSICIONA EL CURSOR
13580 PRINT "PARA TERMINAR,A LA SOLICITUD DEL
      NOMBRE DEL CLIENTE RESPONDA 'zzzz'"
13590 GOSUB 10000:REM -->MASCARA
13600 XPOS=13:YPOS=4
13610 SW=3
13620 GOSUB 1000:REM --->TECLADO
13630 NOMINATIVO$=ALFA$
13640 IF NOMINATIVO$="zzzz" THEN RETURN
13650 CLIENTE$=NOMINATIVO$
13660 SW=3
13670 XPOS=13
13680 YPOS=6
13690 GOSUB 1000:REM --->TECLADO
13700 DIRECCION$=ALFA$
13710 SW=3
13720 XPOS=13
13730 YPOS=8
13740 GOSUB 1000:REM --->TECLADO
13750 CIUDAD$=ALFA$
13760 SW=2
13770 XPOS=13
13780 YPOS=10
13790 GOSUB 1000:REM --->TECLADO
13800 TELEFONO#=GRANDE#
13810 SW=3
```



```
13820 XPOS=13
13830 YPOS=12
13840 GOSUB 100:REM --->TECLADO
13850 PREFIJO$=ALFA$
13860 TIPO$=" "
13870 RETURN
15000 REM ::::::::::::::::::::::::::::::::
15010 REM ::  LECTURA FICHERO CLIENTES  ::
15020 REM ::::::::::::::::::::::::::::::::
15030 REM
15040 INPUT #1,CL$,IN$,CI$,TE$,PR$
15050 RETURN
15090 REM
15500 REM ::::::::::::::::::::::::::::::::::
15510 REM ::  ESCRITURA FICHERO CLIENTES  ::
15520 REM ::::::::::::::::::::::::::::::::::
15530 REM
15550 WRITE #1,CL$,IN$,CI$,TE$,PR$
15560 RETURN
15590 REM
15800 REM ::::::::::::::::::::::::::
15810 REM ::  POSICIONA EL CURSOR  ::
15820 REM ::::::::::::::::::::::::::
15830 REM
15840 LOCATE YPOS,XPOS,1
15850 RETURN
15860 REM
15900 REM :::::::::::::::::::::::::::::
15910 REM ::  LECTURA DE UN CARACTER  ::
15920 REM :::::::::::::::::::::::::::::
15930 REM
15940 CAR$=INPUT$(1)
15950 RETURN
15960 REM
```

APENDICE B

APENDICE PASCAL

GESTION SECUENCIAL

EN PASCAL

```
PROGRAM sec_gestion;
TYPE    cli=RECORD
                cliente   :STRING[31];
                direccion :STRING[33];
                ciudad    :STRING[15];
                telofono  :INTEGER[8];
                prefijo   :STRING[4] ;
        END;
VAR
        clientes     :FILE OF cli;
        actualizacion:FILE OF cli;
        fuera,
        fin          :BOOLEAN;
        partida,
        termino,
        alfa,
        nominativo   :STRING[33];
        car,sw,
        respuesta,
        que_quiere   :CHAR;
        st           :STRING[1];
```

```
       grande      :INTEGER[15];
       x,y,z,j,
       x_pos,y_pos,
       inicio,alt,
       pequeno:INTEGER;

PROCEDURE borrado_pantalla;
  BEGIN
    WRITE(CHR$(28));
  END;

PROCEDURE teclado;
BEGIN
  CASE sw OF
       '1':pequeno:=0;
       '2':grande:=0;
       '3':alfa:=0;
  END;
  GOTOXY(x_pos,y_pos);
  REPEAT
    READ(KEYBOARD,car);
    IF NOT EOLN(KEYBOARD)
       THEN BEGIN IF ORD(car)=8 THEN BEGIN
              GOTOXY(x_pos,y_pos);
              CASE sw OF
                 '1':BEGIN
                      pequeno:=pequeno DIV 10;
                      WRITE(pequeno,' ');
                      GOTOXY(x_pos,y_pos);
                      WRITE(PEQUENO);
                    END;
                 '2':BEGIN
                      grande:=grande DIV 10;
                      WRITE(grande,' ');
                      GOTOXY(x_pos,y_pos);
                      WRITE(grande);
                    END;
                 '3':BEGIN
                      DELETE(alfa,LENGTH(alfa),1);
                      WRITE(alfa,' ');
                      GOTOXY(x_pos,y_pos);
```



```
                           WRITE (alfa);
                         END;
                    END;
                   END
                   ELSE BEGIN
                    IF sw  IN ['1','2']
                       THEN BEGIN
                       IF car IN['0'..'9']
                         THEN BEGIN
                               IF sw='1'
                                THEN pequeno:=pequeno*10+ORD(car)-48
                                ELSE grande:=grande*10+ORD(car)-48;
                                write
                              END
                         ELSE WRITE(CHR$(7));
                            END  {del campo numerico}
                       ELSE BEGIN
                             IF car IN [' '..'"']
                               THEN BEGIN
                                      st:=' ';
                                      st[1]:=car;
                                      alfa:=CONCAT(alfa,st);
                                      WRITE(car);
                                     END
                                     ELSE WRITE(CHR$(7));
                            END;   {del campo alfabetico}
                         END;  {si no tecla de retorno}
                     END;   {no es <RETURN>}
                 UNTIL EOLN(KEYBOARD);
             END;

PROCEDURE menu;
BEGIN
  borrado_pantalla;
  WRITELN;
  WRITELN('1 --> CARGA DE DATOS');
  WRITELN;
  WRITELN('2 --> LECTURA CON EL NOMBRE DE CLIENTE');
  WRITELN;
  WRITELN('3 --> LECTURA CON EL NUMERO DE REGISTRO');
  WRITELN;
```

```
  WRITELN('4 --> MODIFICACION O ANULACION DE REGISTRO');
  WRITELN;
  WRITELN('5 --> PUESTA EN COLA DEL REGISTRO AL FINAL');
  WRITELN;
  WRITELN('9 --> FIN DEL PROGRAMA');
  WRITELN;
  READ(QUE_QUIERE);WRITELN;
END;

PROCEDURE visualizacion;
BEGIN
 WITH CLIENTE^
 DO BEGIN
      GOTOXY(13,4);
      WRITE(cliente^.cliente);
      GOTOXY(13,6);
      WRITE(direccion);
      GOTOXY(13,8);
      WRITE(ciudad);
      GOTOXY(13,10);
      WRITE(telefono);
      GOTOXY(13,12);
      WRITE(prefijo);
   END;
END;

PROCEDURE mascara;
BEGIN
     GOTOXY(0,4);
     WRITE('cliente     :');
     GOTOXY(0,6);
     WRITE('direccion   :');
     GOTOXY(0,8);
     WRITE('CIUDAD      :');
     GOTOXY(0,10);
     WRITE('telefono    :');
     GOTOXY(0,12);
     WRITE('prefijo     :');
   END;
```



```
PROCEDURE clave;
BEGIN
 WITH cliente^
 DO REPEAT
    IF cliente<>nominativo THEN GET (cliente);
    UNTIL  (cliente=nominativo)
     OR EOF(cliente);
    fin:=EOF(cliente);
END;

PROCEDURE modificacion;
BEGIN
 RESET  (cliente,'cliente.dat');
 REWRITE(actualizacion,'cliente.nov');
 WHILE NOT EOF(cliente)
 DO BEGIN
   borrado_pantalla;
   mascara;
   visualizacion;
   GOTOXY(0,20);
   WRITE('¿M)odificacion,C)ancelar,R)ecopia?:');
   READ(respuesta);
   IF respuesta IN ['C','c']
     THEN BEGIN   {BORRADO}
       GOTOXY(0,20);
       WRITE('¿QUIERE BORRARLO? (S/N)');
       READ(respuesta);
       IF respuesta IN ['N','n']
        THEN BEGIN
             actualizacion^:=cliente^;
             PUT (actualizacion);
              END;
          END;
     ELSE IF respuesta IN ['M','m']
      THEN BEGIN       {modificacion}
       GOTOXY(0,13);
       WRITE('PULSE <ESC> PARA MODIFICAR EL CAMPO,
              <RETURN> PARA SEGUIR');
       GOTOXY(13,4);
       READ(respuesta);
       IF respuesta=CHR$(27)
```

```
  THEN BEGIN
  GOTOXY(13,4);
  WRITE('              ');
  x_pos:=13;
  y_pos:=4;
  sw:='3';
  cliente^.cliente:=alfa;
END;

  GOTOXY(13,6);
  READ(respuesta);
  IF respuesta=CHR$(27)
  THEN BEGIN
     GOTOXY(13,6);
     WRITE('                                   ');
     x_pos:=13;
     y_pos:=6;
     teclado;
     cliente^.direccion:=alfa;
   END;

  GOTOXY(13,8);
  READ(respuesta);
  IF respuesta=CHR$(27)
  THEN BEGIN
     GOTOXY(13,8);
     WRITE('              ');
     x_pos:=13;
     y_pos:=8;
     teclado;
     cliente^.ciudad:=alfa;
   END;

  GOTOXY(13,10);
  READ(respuesta);
  IF respuesta=CHR$(27)
  THEN BEGIN
     GOTOXY(13,10);
     WRITE('              ');
     x_pos:=13;
     y_pos:=10;
```



```
    sw:='2';
    teclado;
    cliente^.telefono:=grande;
  END;

 GOTOXY(13,12);
 READ(respuesta);
 IF respuesta=CHR$(27)
 THEN BEGIN
    GOTOXY(13,12;
    WRITE('              ');
    x_pos:=13;
    y_pos:=12
    sw:='3';
    teclado;
    cliente^.prefijo:=alfa;
  END;
    actualizacion^:=cliente^;
    PUT (actualizacion);
  END    {modificacion}

 ELSE BEGIN  {recopia}
     actualizacion^:=cliente^;
     PUT (actualizacion);
     END;  {inalterado}
     GET (cliente);
   END;  {registro existente}
   CLOSE(cliente);
   CLOSE(actualizacion,LOCK);
 END;

PROCEDURE nombre
BEGIN
 RESET(cliente,'cliente.dat');
 GOTOXY(0,20);
 WRITE('NOMBRE DEL PRIMER CLIENTE   :');
 READLN(partida);
 GOTOXY(0,20);
 WRITE('NOMBRE DE ULTIMO CLIENTE   :',
      '                                 ');
 GOTOXY(32,20);
```

```
 READLN(termino);
 nominativo:=partida;
 clave;
 IF NOT fin
  THEN REPEAT
   borrado_pantalla;
   mascara;
   visualizacion;
   fuera:=cliente^.cliente=termino;
   GOTOXY(0,20);
   GET(cliente);
   WRITE('<ESPACIO> PARA SEGUIR,',
         '<ESC> PARA TERMINAR');
   READ(respuesta);
  UNTIL (respuesta)=CHR$(27)
        OR (fuera)
        OR EOF(cliente);
    CLOSE(cliente);
  END;

PROCEDURE posiciona;
BEGIN
 x:=0;
 REPEAT
  x:=x+1;
  IF x<inicio then GET (cliente);
 UNTIL (x=inicio)
       OR EOF(cliente);
      fin:=EOF(cliente);
 END;

PROCEDURE lectura;
BEGIN
 RESET(cliente,'cliente.dat');
 GOTOXY(0,20);
 WRITE('NUMERO DEL PRIMER REGISTRO:    ');
 sw:='1';
 x_pos:=32;
 y_pos:=20;
 teclado;
 inicio:=pequeno;
```



```
GOTOXY(0,20);
WRITE('NUMERO DEL ULTIMO REGISTRO:   ',
      '                               ');
x_pos:=32;
y_pos:=20;
teclado;
alt:=pequeno;
posiciona;
IF NOT fin
 THEN  REPEAT
   borrado_pantalla;
   mascara;
   visualizacion;
   GOTOXY(0,20);
   GET(cliente);
   inicio:=inicio+1;
   WRITE('<ESPACIO> PARA SEGUIR,',
         '<ESC> PARA TERMINAR');
   READ(respuesta);
  UNTIL  (respuesta=CHR$(27))
       OR (inicio>alt)
       OR (EOF(cliente));
 END;

PROCEDURE solicitudes;
BEGIN
 borrado_pantalla;
 GOTOXY(0,20);
 WRITELN('PARA FINALIZAR LA CARGA,RESPONDA',
        '"zzzz" A LA PETICION DE CLIENTE');
 mascara;
 x_pos:=13;
 y_pos:=4;
 sw:='3';
 teclado;
 nominativo:=alfa;
 IF nominativo<>'zzzz'
  THEN BEGIN
  WITH cliente^
  DO BEGIN
     cliente:=nominativo;
```

```
        x_pos:=13;
        y_pos:=6;
        teclado;
        direccion:=alfa;
        x_pos:=13;
        y_pos:=8;
        teclado;
        ciudad:=alfa;
        x_pos:=13;
        y_pos:=10;
        sw:='2';
        teclado;
        telefono:=grande;
        x_pos:=13;
        y_pos:=12;
        sw:='3';
        teclado;
        prefijo:=alfa;
      END;   {DE LA WITH CLIENTE^}
    END; {SI CLIENTE}
END;

PROCEDURE carga;
BEGIN
  REWRITE(cliente,'cliente.dat');
  REPEAT
  solicitudes;
  IF nominativo<>'zzzz' THEN PUT(cliente);
  UNTIL nominativo='zzzz';
  CLOSE(cliente,LOCK);
END;

PROCEDURE puesta_en_fila;
  RESET(cliente,'cliente.dat);
  REWRITE(actualizazion,'cliente.nov);
  WHILE NOT EOF (cliente)
  DO BEGIN
   actualizacion^:=cliente^;
   WRITELN(cliente^.cliente);
   PUT(actualizacion);
   GET(cliente);
```



```
   END;
  REPEAT
    solicitudes;
  IF nominativo<>'zzzz'
   THEN BEGIN
    actualizacion^:=cliente^;
   PUT(actualizacion);
  END;
 UNTIL nominativo='zzzz';
 CLOSE(cliente);
 CLOSE(actualizacion,LOCK);
END;

BEGIN
  menu;
  CASE que_quiere OF
        '1':            carga;
        '2':           nombre;
        '3':          lectura;
        '4':     modificacion;
        '5':   puesta_en_fila;
 END;
END.
```

ACCESO ALEATORIO

PREPARACION

```
PROGRAM ran_preparacion;
TYPE
        cli=RECORD
             CASE BOOLEAN OF
                TRUE:(cliente:STRING[31];
                      direccion:STRING[33];
                      ciudad:STRING[15];
                      telefono:INTEGER[8];
                      prefijo:STRING[4];
                      tipo:CHAR);
```

```
              FALSE:(primero:integer;
                     maximo:integer);
           END;
 VAR
    cliente:FILE OF cli;
    x,ultimo:INTEGER;
 BEGIN
  REWRITE(cliente,'cliente.dat');
  WRITE('¿DE CUANTOS REGISTROS ESTARA COMPUESTO EL FICHERO?');
  READLN(ultimo);

  WITH cliente^
  DO BEGIN
    primero:=2;
    maximo:=ultimo+1;
    PUT(cliente);
    PUT(cliente);
    cliente:='';
    direccion:='';
    ciudad:='';
    telefono:=0;
    prefijo:='0000';
      tipo:='';
      FOR X:=2 TO ultimo+1 DO PUT (cliente);
    END;
    CLOSE
 END.
```

ACCESO ALEATORIO

GESTION

```
PROGRAM ran_gestion;
TYPE
        cli=RECORD
             CASE BOOLEAN OF
                TRUE:(cliente:STRING[31];
                      direccion:STRING[33];
```



```
                      ciudad:STRING[15];
                      telefono:INTEGER[8];
                      prefijo:STRING[4];
                      tipo:CHAR);
                FALSE:(primero:integer;
                       maximo:integer);
             END;

VAR
        cliente:FILE OF cli;
        fuera,fin:BOOLEAN;
        partida,termino,alfa,nominativo:STRING[33]
        car,sw,respuesta,que_quiera:CHAR;
        st:STRING[1];
        grande:INTEGER[15];
        x,y,z,j,
        x_pos,y_pos,
        inicio,alt,
        pequeno,actual,ultimo:INTEGER;

PROCEDURE borrado_pantalla;
BEGIN
  WRITE(CHR$(28));
END;

PROCEDURE teclado;
BEGIN
  CASE sw OF
        '1':pequeno:=0;
        '2':grande:=0;
        '3':alfa:='';
  END;
  GOTOXY(x_pos,y_pos);
  REPEAT
    READ(KEYBOARD,car);
    IF NOT EOLN(KEYBOARD)
       THEN BEGIN
             IF ORD(car)=8 THEN BEGIN
             GOTOXY(x_pos,y_pos);
             CASE sw OF
                '1':BEGIN
```

```
               pequeno:=pequeno DIV 10;
               WRITE(pequeno,' ');
               GOTOXY(x_pos,y_pos);
               WRITE(PEQUENO);
             END;
          '2':BEGIN
               grande:=grande DIV 10;
               WRITE(grande,' ');
               GOTOXY(x_pos,y_pos);
               WRITE(grande);
             END;
          '3':BEGIN
               DELETE(alfa,LENGTH(alfa),1);
               WRITE(alfa,' ');
               GOTOXY(x_pos,y_pos);
               WRITE (alfa);
             END;
     END;
    END
    ELSE BEGIN
     IF sw  IN ['1','2']
        THEN BEGIN
        IF car IN['0'..'9']
          THEN BEGIN
               IF sw='1'
               THEN pequeno:=pequeno*10+ORD(car)-48
               ELSE grande:=grande*10+ORD(car)-48;
                 WRITE
                END
          ELSE WRITE(CHR$(7));
             END  {del campo numerico}
        ELSE BEGIN
        IF car IN [' '..'"']
          THEN BEGIN
                 st:=' ';
                 st[1]:=car;
                 alfa:=CONCAT(alfa,st);
                 WRITE(car);
                END
                ELSE WRITE(CHR$(7));
      END;    {del campo alfabetico}
```



```
                 END;  {si no tecla de retorno}
              END;   {no es <RETURN>}
           UNTIL EOLN(KEYBOARD);
        END;


PROCEDURE menu;
BEGIN
  borrado_pantalla;
  WRITELN;
  WRITELN('1 --> CARGA DE DATOS');
  WRITELN;
  WRITELN('2 --> LECTURA DE REGISTRO');
  WRITELN;
  WRITELN('4 --> MODIFICACION DE UN REGISTRO');
  WRITELN;
  WRITELN('5 --> ANULACION DE UN REGISTRO')
  WRITELN;
  WRITELN('9 --> FIN DEL PROGRAMA');
  WRITELN;
  READ(QUE_QUIERE);WRITELN;
END;


PROCEDURE visualizacion;
BEGIN
 WITH cliente^
 DO BEGIN
       GOTOXY(13,4);
       WRITE(cliente^.cliente);
       IF TIPO <>' 'THEN WRITE('ANULADO')
       ELSE WRITE('             ')
       GOTOXY(13,6);
       WRITE(direccion);
       GOTOXY(13,8);
       WRITE(ciudad);
       GOTOXY(13,10);
       WRITE(telefono);
       GOTOXY(13,12);
       WRITE(prefijo);
    END;
END;
```

```
 PROCEDURE mascara;
 BEGIN
   GOTOXY(0,4);
   WRITE('cliente      :');
   GOTOXY(0,6);
   WRITE('direccion    :');
   GOTOXY(0,8);
   WRITE('CIUDAD       :');
   GOTOXY(0,10);
   WRITE('telefono     :');
   GOTOXY(0,12);
   WRITE('prefijo      :');
 END;

 PROCEDURE lista;
 BEGIN
  fin:=FALSE;
  IF (inicio<actual)
      AND (inicio>0)
  THEN BEGIN
   SEEK(cliente,inicio);
   GET(cliente);
   inicio:=inicio+1;
  END
  ELSE fin:=TRUE;
 END;

 PROCEDURE modificacion;
 BEGIN
  borrado_pantalla;
  GOTOXY(0,20);
  WRITE('NUMERO DEL REGISTRO:');
  sw:'1';
  x_pos:=18;
  y_pos:=20;
  teclado;
  inicio:=pequeno+1;
  j:=pequeno+1;
  lista;
  IF NOT fin
  THEN BEGIN
```



```
  mascara;
  visualizacion;
  GOTOXY(0,20);
  WRITE('<ESC> PARA MODIFICAR,',
        '<RETURN> PARA CONTINUAR');
  GOTOXY(13,4);
  READ(respuesta);
  IF respuesta=CHR$(27)
  THEN BEGIN
  GOTOXY(13,4);
  WRITE('                ');
  sw:='3':
  x_pos:=13;
  y_pos:=4;
  teclado;
  cliente^.cliente:=alfa;
 END;

 GOTOXY(13,6);
 READ(respuesta);
 IF respuesta=CHR$(27)
  THEN BEGIN
   GOTOXY(13,6);
   WRITE('                          ');
   sw:='3';
   x_pos:=13;
   y_pos:=6;
   teclado;
   cliente^.direccion:=alfa;

  GOTOXY(13,8);
  READ(respuesta);
  IF respuesta=CHR$(27)
   THEN BEGIN
    GOTOXY(13,8);
    WRITE('                          ');
    sw:='3';
    x_pos:=13;
    y_pos:=8;
    teclado;
    cliente^.ciudad:=alfa;
```

```
 GOTOXY(13,10);
 READ(respuesta);
 IF respuesta=CHR$(27)
  THEN BEGIN
   GOTOXY(13,10);
   WRITE('                              ');
   sw:='1';
   x_pos:=13;
   y_pos:=10;
   teclado;
   cliente^.telefono:=pequeno;

 GOTOXY(13,12);
 READ(respuesta);
 IF respuesta=CHR$(27)
  THEN BEGIN
   GOTOXY(13,12);
   WRITE('                              ');
   sw:='3';
   x_pos:=13;
   y_pos:=12;
   teclado;
   cliente^.prefijo:=alfa;
 END;
 SEEK(cliente,j);
 PUT(cliente);
 END
   ELSE BEGIN
     GOTOXY(2,20);
     WRITELN('NO EXISTE ESTE REGISTRO,',
             'PULSE <RETURN>');
     READLN;
    END;
END;

PROCEDURE borra;
BEGIN
 borrado_pantalla;
 GOTOXY(0,20);
 WRITE('NUMERO DEL REGISTRO:  ');
 sw:='1';
```



```
  x_pos:=18;
  y_pos:=20;
  teclado;
  inicio:=pequeno+1;
  j:=pequeno+1;
  lista;
  IF NOT fin
  THEN BEGIN
   mascara;
   visualizacion;
   GOTOXY(0,20);
   WRITE('¿QUIERE BORRARLO?(S/N)');
   READ(respuesta);
   IF respuesta IN['S','s']
    THEN BEGIN
      cliente^.tipo:='C';
      SEEK(cliente,j);
      PUT(cliente);
    END;
 END
 ELSE BEGIN
    GOTOXY(0,20);
    WRITELN('NO EXISTE ESTE REGISTRO,',
            'PULSE <RETURN>');
    READLN;
  END;
END;

PROCEDURE lectura
BEGIN
GOTOXY(0,20);
 WRITE('NUMERO DEL PRIMER REGISTRO:    ');
 x_pos:=32;
 y_pos:=20;
 sw:='1';
 teclado;
 inicio:=pequeno+1;
 GOTOXY(0,20);
 WRITE('NUMERO DEL ULTIMO REGISTRO:    ',
       '                                ');
 x_pos:=32;
```

```
 y_pos:=20;
 sw:='1';
 teclado;
 alt:=pequeno+1;
 IF alt>actual-1 THEN alt:=actual-1;
 lista;
 IF NOT fin
 THEN REPEAT
   borrado_pantalla;
   mascara;
   visualizacion;
   GOTOXY(0,20);
   lista;
   WRITE('<ESPACIO> PARA CONTINUAR,',
        '<ESC> PARA ACABAR');
   READ(respuesta);
  UNTIL (respuesta=CHR$(27))
       OR (inicio>alt+1)
       OR (fin);
END;

PROCEDURE carga;
BEGIN
 borrado_pantalla;
 WRITELN('LA CARGA PARTE CON EL NUMERO',actual-1);
 IF actual>2
  THEN BEGIN
   SEEK(cliente,PRED(actual)) ;
   GET(cliente);
   WRITELN('EL REGISTRO ANTERIOR',
          'ESCRITO ES EL SIGUIENTE')
   mascara;
   visualizacion;
   WRITELN;
   GOTOXY(0,20);
   WRITE('PULSE <RETURN> PARA CONTINUAR');
END;

REPEAT
borrado_pantalla;
GOTOXY(0,20);
```



```
WRITELN('PARA TERMINAR RESPONDA "zzzz"',
        'A LA SOLICITUD DEL NOMBRE DEL CLIENTE');
mascara;
sw:='3';
x_pos:=13;
y_pos:=4;
teclado;
nominativo:=alfa;
IF nominativo<>'zzzz'
 THEN BEGIN
   IF actual>ultimo
     THEN BEGIN
       WRITE('NO PUEDO CONTINUAR,SE HA SUPERADO',
             'LA DIMENSION MAXIMA.PULSE<RETURN>');
       READLN;
       EXIT(carga);
    END;
calculo;
busqueda;
IF NOT fin
  THEN BEGIN
    WRITE('ESTE CLIENTE YA EXISTE, NO PUEDO',
          'INCLUIRLO. PULSE <RETURN>');
    READLN;
    EXIT(carga);
END;
WITH cliente^
DO BEGIN
 cliente:=nominativo;
 x_pos:=13;
 y_pos:=6
 teclado;
 direccion:=alfa;
 y_pos:=8;
 teclado;
 ciudad:=alfa;
 sw:='2';
 y_pos:=10;
 teclado;
 telefono:=grande;
 sw:='3';
```

```
  y_pos:=12;
  teclado;
  prefijo:=alfa;
  tipo:=' ';
  SEEK(indice,hash);
  GET(indice)
  colision:=indice^
  indice^:=actual;
  SEEK(indice,hash);
  PUT(indice);
        SEEK(cliente,actual);
        PUT(cliente);
        actual:=succ(actual);
        primero:=actual;
        maximo:=ultimo;
        SEEK(cliente,1);
        PUT(CLIENTE);
   END; {DE LA WITH CLIENTE^}
 END; {SI CLIENTE}
UNTIL nominativo='zzzz'
END;

BEGIN
 RESET (cliente,'cliente.dat');
 SEEK(cliente,1) ;
 GET(cliente);
 actual:=cliente^.primero;
 ultimo:=cliente^.maximo;
 REPEAT
  menu;
   CASE que_quiere OF
        '1':         carga;
        '2':       lectura;
        '4':  modificacion;
        '5':       borrado;
   END;
 UNTIL que_quiere='9';
 CLOSE(cliente,LOCK);
END.
```


ACCESO ALEATORIO

COMPACTACION

```
PROGRAM ran_compactacion;
TYPE
        cli=RECORD
             CASE BOOLEAN OF
                TRUE:(cliente:STRING[31];
                       direccion:STRING[33];
                       ciudad:STRING[15];
                       telefono:INTEGER[8];
                       prefijo:STRING[4];
                       tipo:CHAR);
                FALSE:(primero:integer;
                        maximo:integer);
             END;

VAR
        viejos,cliente:FILE OF cli;
        x,y,z,actual,ultimo:INTEGER;

BEGIN
  RESET(viejos,'cliente.vie');
  RESET(cliente,'cliente.dat');
  SEEK(cliente,1);
  GET(cliente);
  ultimo:=cliente^.maximo;
  actual:=viejos^.primero;
  GET(viejos);
  y:=2
  z:=2;
  REPEAT
   WITH viejos^
DO BEGIN
  IF tipo='C'
     THEN WRITELN(cliente:15,'ANULADO');
     ELSE BEGIN
       WRITELN(cliente:15);
        cliente^:=viejos;
```

```
        SEEK(cliente,z);
        PUT(cliente);
        z:=z+1;
     END;
     GET(viejos);
     y:=y+1;
     IF z-1>ultimo
       THEN BEGIN
          WRITELN('HAY ',z,' REGISTROS ACTIVOS,',
                 'EN UN FICHERO QUE LOS PREVEE',
                 ultimo,'!');
          WRITELN('NO PUEDO SEGUIR.');
          WRITELN('PULSE <RETURN>');
          READLN;
          EXIT(program);
       END;
     END;  {DE LA WITH CLIENTE^};
  UNTIL y>=actual;
  SEEK(cliente,1);
  GET(cliente);
  cliente^.primero:=z;
  SEEK(cliente,1);
  PUT(cliente);
  CLOSE(viejos);
  CLOSE(cliente);
END.
```


BIBLIOGRAFIA

Introducción a la programación. Estructuras de datos.
Clavel-Biondi.
Masson.

Introducción a las bases de datos relacionales.
Mayne-Wood.
Díaz de Santos.

Técnicas de bases de datos.
Campderrich.
Díaz de Santos.

Programación de archivos de datos en BASIC.
Finkel-Brown.
Díaz de Santos.

Diseño de bases de datos.
Wiederhold.
Díaz de Santos.

El Apple y sus ficheros.
J. Boisgontier.
Elisa.

Las bases de datos dBASE II en el ordenador personal IBM.
Daniel Lanzas, B. Labrador, J. A. Jaén.
Universidad Politécnica de Madrid. E.T.S.I.I.

Aplique el dBASE II.
Townsend.
McGraw-Hill.

Using dBASE III.
Jones.
McGraw-Hill.

BIBLIOTECA BASICA INFORMATICA

INDICE GENERAL

10 Sistemas operativos y software de base
Qué son, para qué sirven. Unos desconocidos muy importantes.
11 Sistema operativo CP/M
Uno de los sistemas operativos para microprocesadores de 8 bits de mayor difusión en el mercado.
12 MS-DOS: el estándar de IBM
Sistema operativo para el microprocesador de 16 bits 8088, adoptado por el IBM-PC.
13 Paquetes de aplicaciones. Software "pret a porter"
Características y peculiaridades de los más importantes paquetes de aplicaciones.
14 VisiCalc: una buena hoja de cálculo
Interioridades y manejo de una de las hojas de cálculo más usadas.
15 Dibujar con el ordenador
Profundizando en una de las facetas útiles y divertidas que nos ofrecen los ordenadores.
16 Tratamiento de textos... para escribir con el ordenador
Cómo convertir su ordenador en una máquina de escribir con memoria y todo tipo de posibilidades.
17 Diseño de juegos
Particularidades características de esta aplicación de los ordenadores.
18 LOGO: la tortuga inteligente
Un lenguaje conocido por su «cursor gráfico», la tortuga, y sus aplicaciones pedagógicas al alcance de su mano.
19 BASIC y tratamiento de imágenes
Todo lo que en **¡Y llegó el BASIC!** no se pudo ver sobre las imágenes y gráficos en el BASIC.
20 Bancos de datos (I)
Peculiaridades de una de las aplicaciones de los ordenadores más interesantes, y que más dinero mueven.
21 Bancos de datos (II)
Profundizando en sus características.
22 Paquetes integrados: Lotus 1-2-3 y Simphony
Estudio de dos de los paquetes integrados (Hoja de cálculo + base de datos + ...) más conocidos.
23 dBASE II y dBASE III
Cómo aprovechar las dos versiones más recientes de esta importante base de datos.
24 Los ordenadores uno a uno
Un amplio y completo estudio comparativo.
25 Cálculo numérico en BASIC
Una aplicación especializada a su disposición.
26 Multiplan
Cómo hacer uso de este moderno paquete de aplicaciones.
27 FORTRAN y COBOL
Dos lenguajes muy especializados y distintos.
28 FORTH: anatomía de un lenguaje inteligente
Principales características de un lenguaje moderno, flexible y de amplio uso, en la robótica.
29 Cómo realizar nuestro propio banco de datos
Conocimientos necesarios para poder fabricar un banco de datos a nuestro gusto y medida.
30 Los paquetes integrados uno a uno
Todos los que usted puede encontrar en el mercado.

NOTA: Ingelek, S. A. se reserva el derecho de modificar, sin previo aviso, el orden, título o contenido de cualquier volumen de la colección.

NOTAS



a documentación y el manejo de la información obtenida han sido siempre uno de los grandes problemas informáticos, pues en cuanto el volumen de datos se hacía relativamente importante su proceso manual se hacía lento, costoso y poco eficaz.

La solución se ha materializado en los llamados "bancos de datos", que, independientemente de su medio de almacenamiento, conservan un gran número de datos con una coherencia temática y en las "bases de datos", que nos permiten el acceso, recuperación y actualización de los datos.

En este libro comenzaremos a ver cómo podemos conseguir este tipo de funciones mediante completos programas, presentados en BASIC y en Pascal, que serán detalladamente explicados.

395 pts.
(incluido IVA)